PHÁT TRIỂN HỆ THỐNG ECOMSYS

1. **Mô tả bằng dạng Bảng và ngôn ngữ tự nhiên các chức năng tương ứng với các actor và phi chức năng của Hệ thống ecomSys.**
   A. Mô tả bằng ngôn ngữ tự nhiên.

Trong hệ thống ecomsys , có các chức năng chính sau đây:

- **Khách hàng** có thể đăng kí đăng nhập, quên mật khẩu, thay đổi mật khẩu, thay đổi thông tin cá nhân.
- **Khách hàng** xem sản phẩm, tạo giỏ hàng, đặt hàng.
- **Admin** quản lí khách hàng, đơn hàng, sản phẩm, thanh toán, xem báo cáo.

❖ **Mô tả các actor của hệ thống**

- Primary Actor: Khách hàng – Customer.
  Những người thức hiện, tham gia vào hoạt động mua bán và tím kiếm sản phẩm trên hệ thống, được xem như là người dùng cuối cùng của hệ thống.
  - Hoạt động chính:
    - Thực hiện đăng kí tài khoản và đăng nhập trên hệ thống
    - Xem sản phẩm trên hệ thống.
    - Tìm kiếm các sản phẩm muốn mua.
    - Đặt hàng.
    - Thanh toán.
    - Theo dõi hoặc hủy đơn hàng.
    - Thực hiện đánh giá đơn hàng.
- Primary Actor: Nhân viên hệ thống - Staff.
  Đây là những người thực hiện quản lý vàkiểm tra các đơn hàng trên hệ thống cũng như các dịch vụ liên quan đến khách hàng.
  - Hoạt động chính:
    - Đăng nhập bằng tài khoản nhân viên trên hệ thống.
    - Xác nhận các đơn đặt hàng.
    - Thêm, sửa, xóa thông tin sản phẩm trên hệ thống.
    - Xác nhận hủy đơn hàng khi gặp vấn đề.
    - Xem và tư vấn các đánh giá của khách hàng.
- Secondary Actor: Thu ngân - Bank. Đây là người thực hiện các yêu cầu thanh toán cótrên hệ thống từ thanh toán online hoặc trả tiền mặt. Họ có nhiệm vụ thực hiện thanhtoán và xuất hóa đơn cho khách hàng.
- Secondary Actor: Đơn vị vận chuyển. Họ là những người nhận và vận chuyển đơn hàngtới khách hàng. Có thể thu hộ tiền mặt nếu có yêu cầu.

❖ **Mô tả yêu cầu các chức năng và phi chức năng của hệ thống bằng ngôn ngữ tự nhiên**

- Yêu cầu chức năng
  - Người dùng: đăng nhập,đăng ký

- Nhân viên : đăng nhập,thêm sản phẩm, xác nhận đặt hàng
- Khách hàng: tìm sản phẩm ,xem sản phẩm chi tiết, đặt sản phẩm, trả tiền, thêm sản phẩm,
- Quản lý : thêm nhân viên , thống kê

- Yêu cầu phi chức năng
  - Tính bảo trì: hệ thống nên có thời gian bảo trì ngắn
  - Tính bảo mật: hệ thống bảo mật thông tin của khách hàng
  - Hiệu suất: đảm bảo hiệu suất cao khi có lượng lớn truy nhập vào hệ thống
  - Độ tin cậy hệ thống phải có khả năng sao lưu dữ liệu để tránh bị mất mát dữ liệu
- Mô tả chức năng
  - Khách hàng đặt hàng trên thương mại điện tử : khách hàng đăng nhập theo tài khoản và mật khẩu của mình→ Giao diện của khách hàng hiên ra → Khách hàng vào giỏ hàng của mình mua sản phẩm sẵn có trong giỏ (nếu có) , khách hàng tìm kiếm sản phẩm mà mình muốn mua → Giao diện hiện ra sản phẩm mà khách hàng cần → Khách hàng chọn sản phẩm mình muốn mua, khách hàng có thể mua ngay (nếu muốn) , hoặc cho vào giỏ hàng rồi mua click nút mua (nếu sản phẩm còn trong kho) → Giao diện hiện ra danh sách khách hàng muốn mua ,kèm theo giá của từng loại, tổng tiền phải trả và phương thức xác nhận → Khách hàng kiểm tra lại thông tin tồi click nút xác nhận → Giao diện hiện đặt hàng thành công.

  - Khách hàng hủy đơn đặt hàng : khách hàng đăng nhập theo tài khoản và mật khẩu của mình → Giao diện của khách hàng hiện ra → Khách hàng vào mục đơn hàng của bạn → Giao diện hiện ra thông tin của đơn hàng và trạng thái của đơn hàng , nếu đơn hàng chưa vận chuyển thì có thể click nút hủy đơn hàng , (nếu đơn hàng đã vận chuyển thì nút hủy đơn hàng không xuất hiện)→ Giao diện thông báo hủy đơn hàng thành công
  - Nhân viên thêm sản phẩm mới :Nhân viên đăng nhập vào hệ thống với tài khoản và mật khẩu của mình→ Giao diện của nhân viên hiện ra các chức năng thêm sản phẩm, xác nhận đặt hàng ,phản hồi nhận xét của khách hàng → Nhân viên chọn thêm sản phẩm → Giao diện ra form điền thêm sản phẩm (mã sp, tên sp, giá,hình ảnh minh họa) (sản phẩm không có mã trùng với sản phẩm có trên hệ thống) → Nhân viên điền thông tin đầy đủ và đến khi hết sản phẩm cần thêm và click vào nút xác nhận → Giao diện hiện thông báo thêm sản phẩm thành công

  - Nhân viên xác nhận đơn hàng : Nhân viên đăng nhập vào hệ thống với tài khoản và mật khẩu của mình → Giao diện của nhân viên hiện ra các chức năng thêm sản phẩm, xác nhận đặt hàng ,phản hồi nhận xét của khách hàng→ Nhân viên chọn xác nhận đặt hàng → Giao diện hiện ra danh sách các đơn hàng đã đặt của khách hàng → Nhân viên click nút xác nhận cho

từng đơn hàng và cập nhật dữ liệu để khách hàng dễ dàng theodõi vận chuyển đơn hàng của mình.

- Quản lý thêm nhân viên : Quản lý đăng nhập vào hệ thống với tài khoản và mật khẩu của mình→ Giao diện của quản lý hiện ra : thêm nhân viên , xem báo cáo → Quản lý chọn thêm nhân viên→ Giao diện hiện ra form điền thông tin (mã nv, họ tên, năm sinh, dân tộc, địa chỉ, sdt) → Quản lý điền thông tin theo yêu cầu và click xác nhận → Giao diện thông báo thêm nhân viên thành công và hiện lên tài khoản và mật khẩu của nhân viên để đăng nhập vào hệ thống.

- Quản lý thống kê doanh số bán hàng :Quản lý đăng nhập vào hệ thống với tài khoản và mật khẩu của mình→ Giao diện của quản lý hiện ra : thêm nhân viên , xem báo cáo → Quản lý xem báo cáo→ Giao diện hiện lên ô text nhập ngày bắt đầu và kết thúc → Quản lý nhập ngày bắt đầu và kết thúc và click tìm kiếm → Giao diện hiện ra doanh số bán hàng tổng và của từng loại mặt hàng , click in báo cáo (nếu muốn)→ Nhân viên click vào một hàng để xem chi tiết từng loại sản phẩm của mặt hàng đó

❖ Mô tả dạng bảng

❖ Usecase	Đặt hàng
Actor	Khách hàng
Tiền điều kiện	Tài khoản và mật khẩu
Hậu điêu kiện	Đặt hàng thành công
Hoạt động	Khách hàng đặt hàng trên thương mại điện tử : khách hàng đăng nhập theo tài khoản và mật khẩu của mình→ Giao diện của khách hàng hiên ra → Khách hàng vào giỏ hàng của mình mua sản phẩm sẵn có trong giỏ (nếu có) , khách hàng tìm kiếm sản phẩm mà mình muốn mua → Giao diện hiện ra sản phẩm mà khách hàng cần → Khách hàng chọn sản phẩm mình muốn mua, khách hàng có thể mua ngay (nếu muốn) , hoặc cho vào giỏ hàng rồi mua click nút mua (nếu sản phẩm còn trong kho) → Giao diện hiện ra danh sách khách hàng muốn mua ,kèm theo giá của từng loại, tổng tiền phải trả và phương thức xác nhận → Khách hàng kiểm tra lại thông tin tồi click nút xác nhận → Giao diện hiện đặt hàng thành công.

Usecase	Hủy đơn hàng
Actor	Khách hàng
Tiền điều kiện	Tài khoản và mật khẩu
Hậu điêu kiện	Hủy đơn hàng thành công

Hoạt động	Khách hàng hủy đơn đặt hàng : khách hàng đăng nhập theo tài khoản và mật khẩu của mình → Giao diện của khách hàng hiện ra → Khách hàng vào mục đơn hàng của bạn → Giao diện hiện ra thông tin của đơn hàng và trạng thái của đơn hàng , nếu đơn hàng chưa vận chuyển thì có thể click nút hủy đơn hàng , (nếu đơn hàng đã vận chuyển thì nút hủy đơn hàng không xuất hiện)→ Giao diện thông báo hủy đơn hàng thành công

Usecase	Thêm sản phẩm mới
Actor	Nhân viên
Tiền điều kiện	Tài khoản và điều kiện
Hậu điều kiện	Thêm sản phẩm thành công
Hoạt động	Nhân viên thêm sản phẩm mới :Nhân viên đăng nhập vào hệ thống với tài khoản và mật khẩu của mình→ Giao diện của nhân viên hiện ra các chức năng thêm sản phẩm, xác nhận đặt hàng ,phản hồi nhận xét của khách hàng → Nhân viên chọn thêm sản phẩm → Giao diện ra form điền thêm sản phẩm (mã sp, tên sp, giá,hình ảnh minh họa) (sản phẩm không có mã trùng với sản phẩm có trên hệ thống) → Nhân viên điền thông tin đầy đủ và đến khi hết sản phẩm cần thêm và click vào nút xác nhận → Giao diện hiện thông báo thêm sản phẩm thành công

Usecase	Xác nhận đơn hàng
Actor	Nhân viên
Tiền điều kiện	Tài khoản và mật khẩu
Hậu điều kiện	Xác nhận thành công
Hoạt động	Nhân viên xác nhận đơn hàng : Nhân viên đăng nhập vào hệ thống với tài khoản và mật khẩu của mình → Giao diện của nhân viên hiện ra các chức năng thêm sản phẩm, xác nhận đặt hàng ,phản hồi nhận xét của khách hàng→ Nhân viên chọn xác nhận đặt hàng → Giao diện hiện ra danh sách các đơn hàng đã đặt của khách hàng → Nhân viên click nút xác nhận cho từng đơn hàng và cập nhật dữ liệu để khách hàng dễ dàng theo dõi vận chuyển đơn hàng của mình.

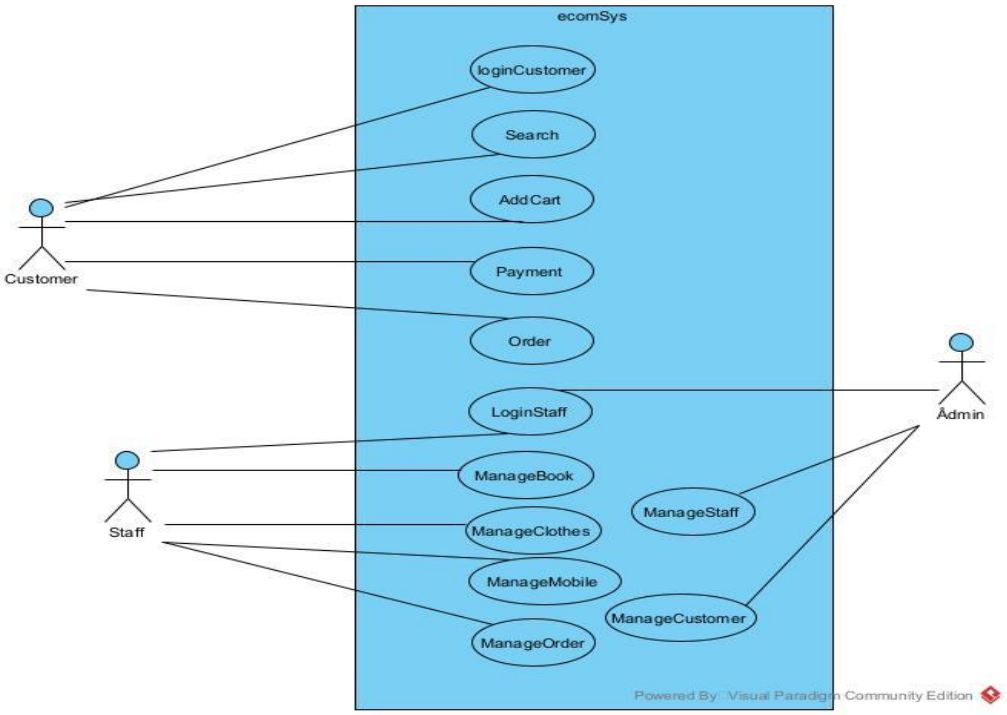
Usecase	Thêm nhân viên
Actor	Quản lý
Tiền điều kiện	Tài khoản và mật khẩu
Hậu điều kiện	Thêm nhân viên thành công

Hoạt động	Quản lý thêm nhân viên : Quản lý đăng nhập vào hệ thống với tài khoản và mật khẩu của mình→ Giao diện của quản lý hiện ra : thêm nhân viên , xem báo cáo → Quản lý chọn thêm nhân viên→ Giao diện hiện ra form điền thông tin (mã nv, họ tên, năm sinh, dân tộc, địa chỉ, sdt) → Quản lý điền thông tin theo yêu cầu và click xác nhận → Giao diện thông báo thêm nhân viên thành công và hiện lên tài khoản và mật khẩu của nhân viên để đăng nhập vào hệ thống.

Usecase	Thống kê
Actor	Quản lý
Tiền điều kiện	Tài khoản và mật khẩu
Hậu điều kiện	Thống kê doang số bán hàng
Hoạt động	Quản lý thống kê doanh số bán hàng :Quản lý đăng nhập vào hệ thống với tài khoản và mật khẩu của mình→ Giao diện của quản lý hiện ra : thêm nhân viên , xem báo cáo → Quản lý xem báo cáo→ Giao diện hiện lên ô text nhập ngày bắt đầu và kết thúc → Quản lý nhập ngày bắt đầu và kết thúc và click tìm kiếm → Giao diện hiện ra doanh số bán hàng tổng và của từng loại mặt hàng , click in báo cáo (nếu muốn)→ Nhân viên click vào một hàng để xem chi tiết từng loại sản phẩm của mặt hàng đó

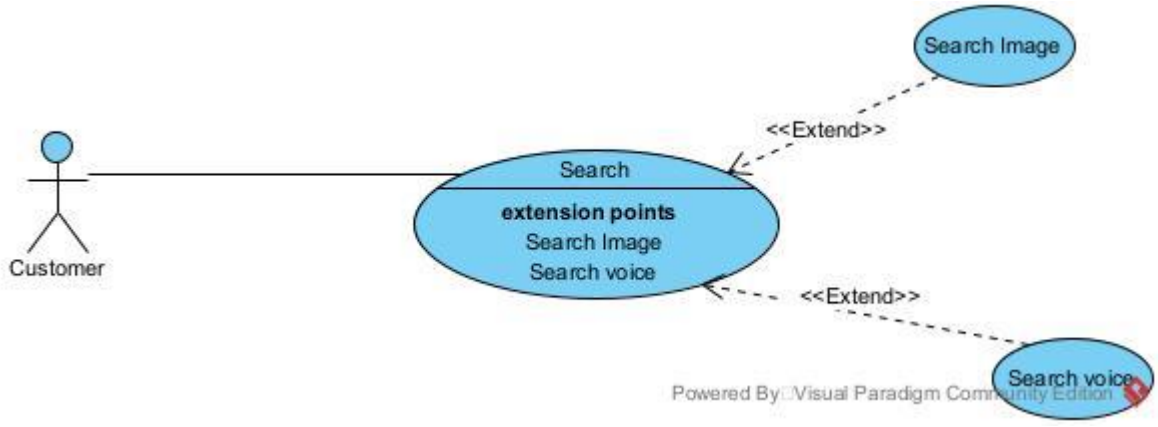
2. **Vẽ Biểu đồ use case tổng quát và biểu đồ usecase chi tiết cho từng chức năng/dịch vụ.**

❖ UseCase tổng quát

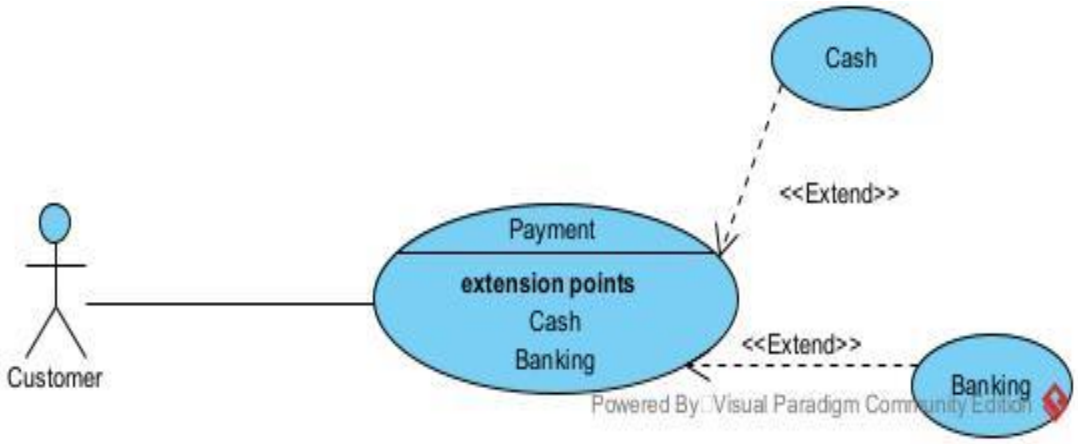


❖ Biểu đồ usecase chi tiết customerLogin:

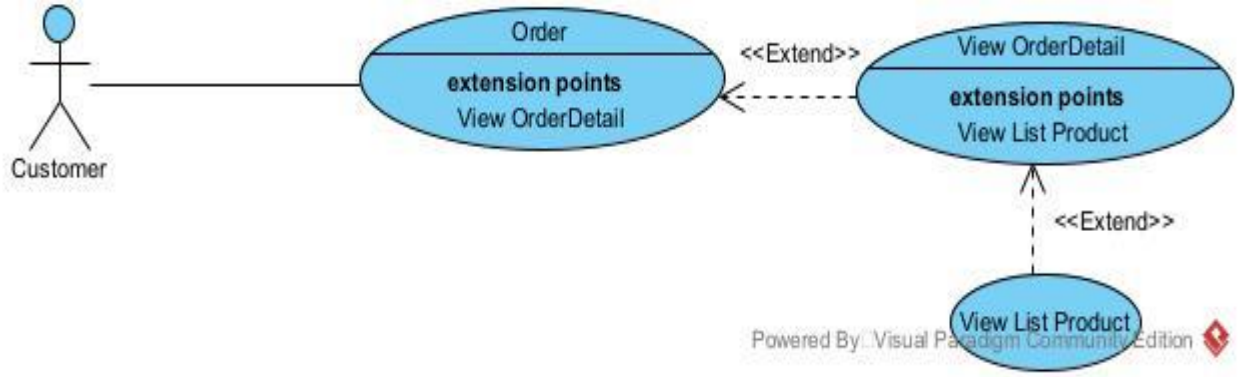
❖ Biểu đồ usecase chi tiết search



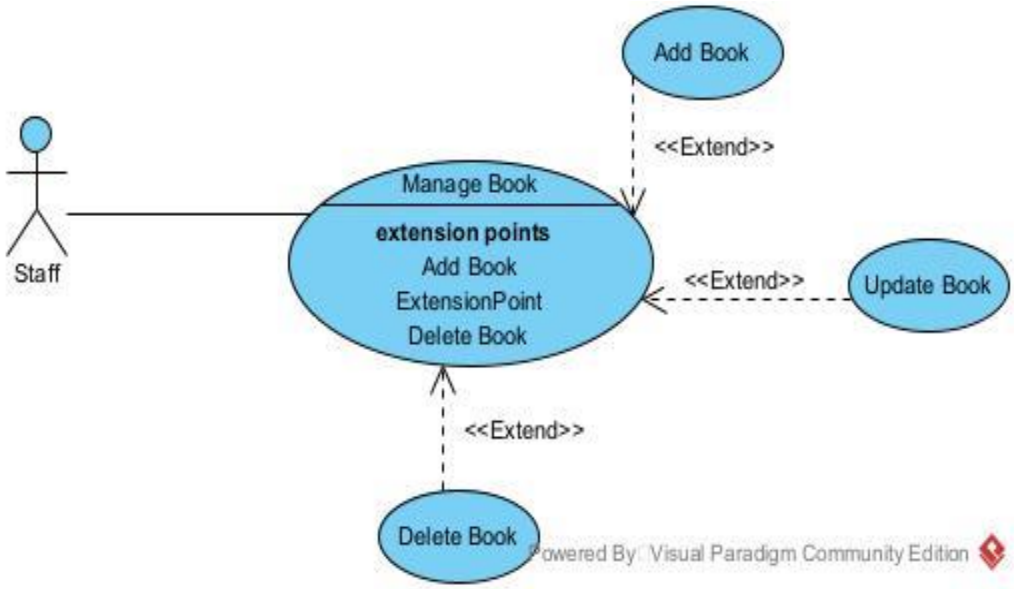
❖ Biểu đồ usecase chi tiết payment



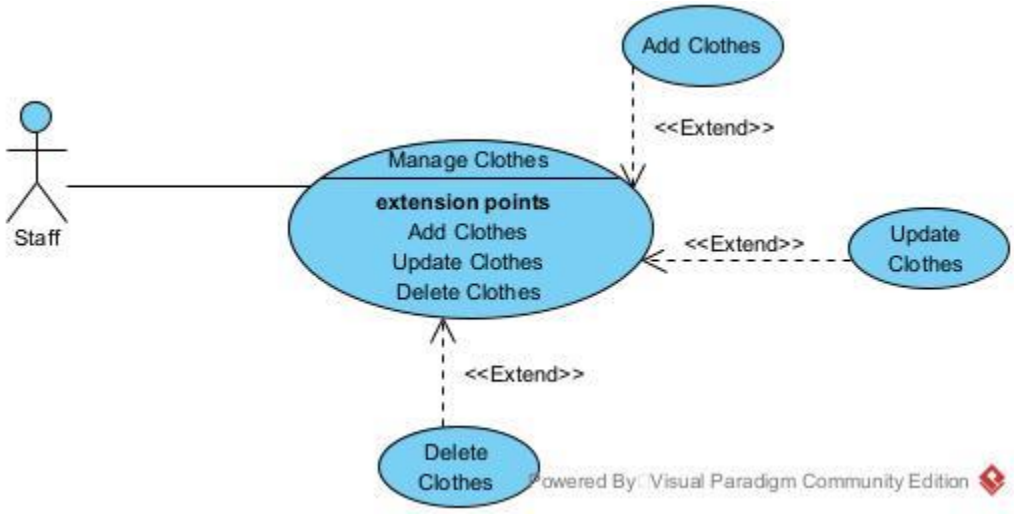
❖ Biểu đồ usecase chi tiết order



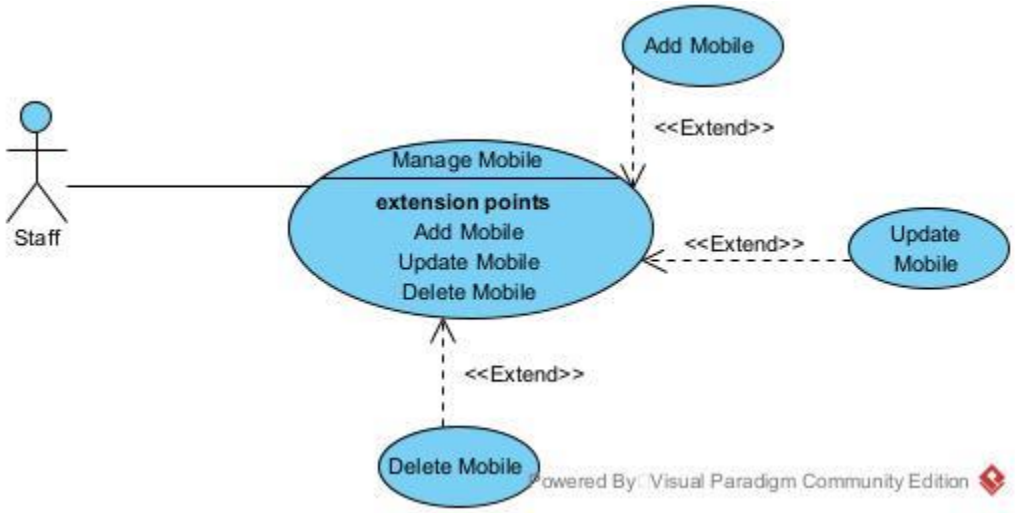
❖ Biểu đồ usecase chi tiết manage Book



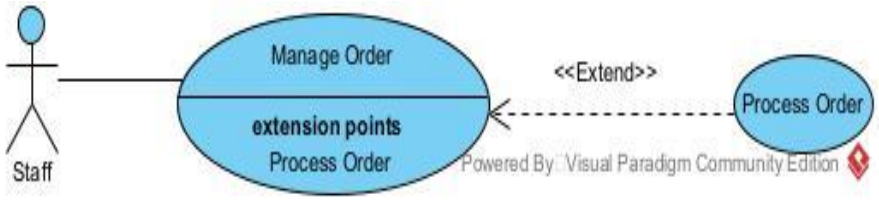
❖ Biểu đồ usecase chi tiết manage Clothes



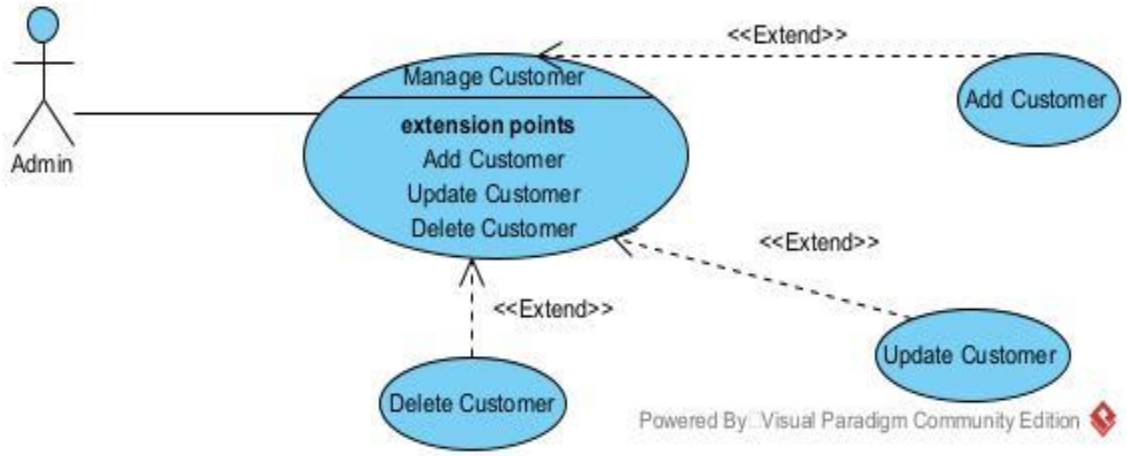
❖ Biểu đồ usecase chi tiết manage Mobile



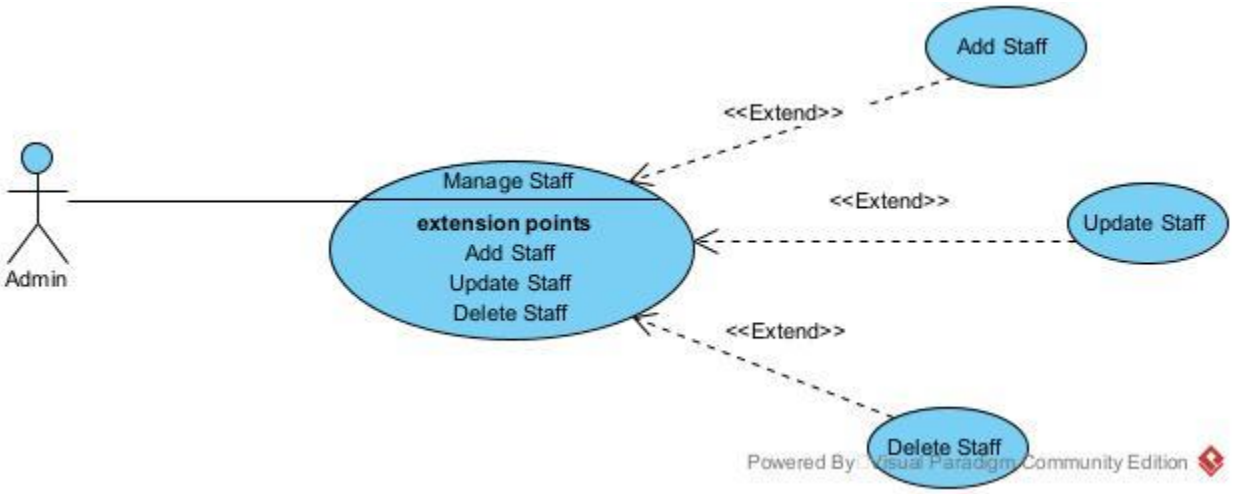
❖ Biểu đồ usecase chi tiết manage order



❖ Biểu đồ usecase chi tiết manage Customer



❖ Biểu đồ usecase chi tiết manage Staff

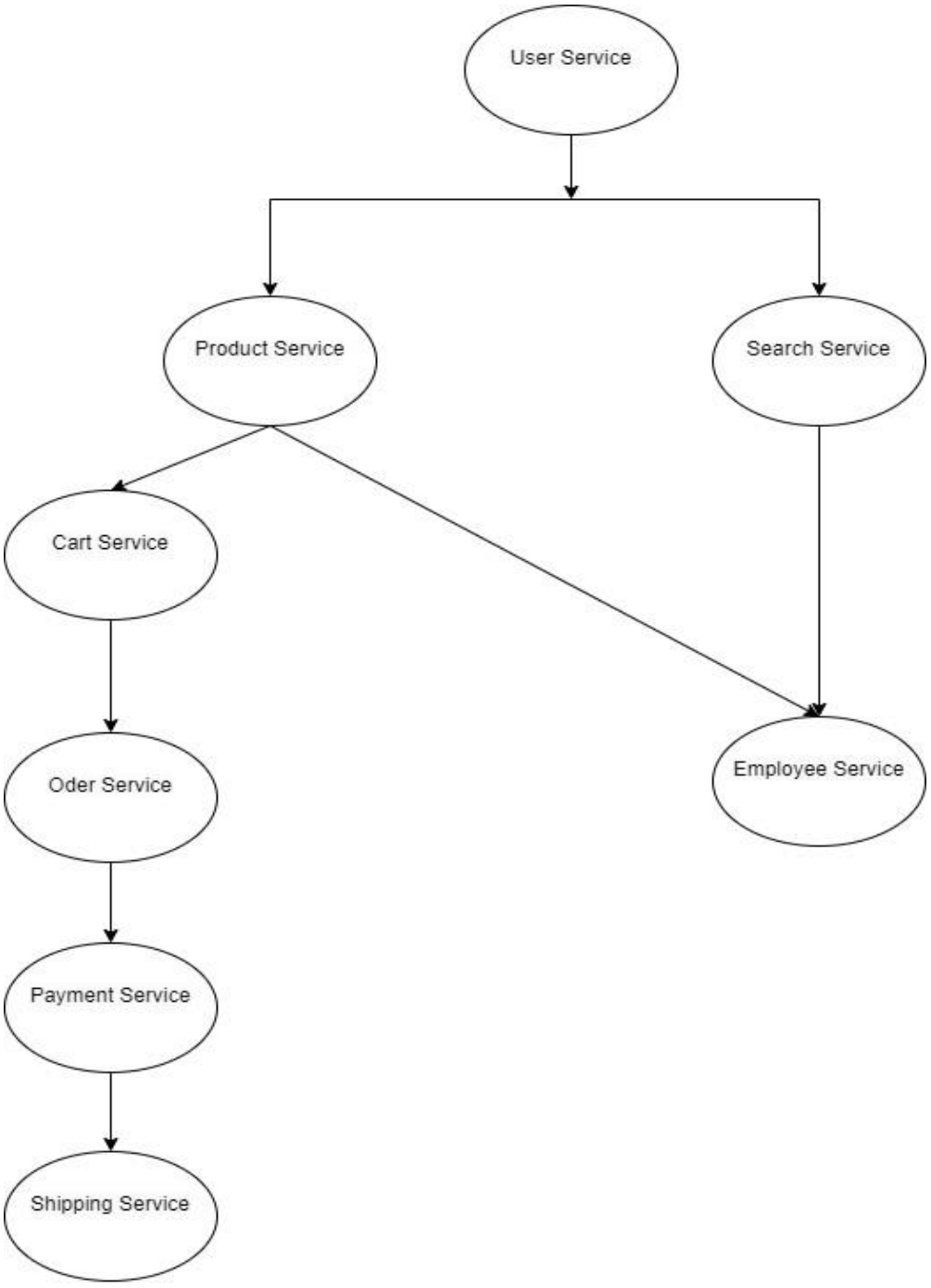


3. **Vẽ biểu đồ phân rã Hệ ecomSys thành các service và các tương tác giữa các dịch vụ (sử dụng quan hệ <> trong UML).**
   ❖ Hệ ecomSys sẽ được phân rã thành các service như sau:

- **Dịch vụ người dùng:** Dịch vụ này liên quan đến việc quản lý tài khoản người dùng và xác thực, chẳng hạn như cho phép người dùng tạo và đăng nhập vào tài khoản của họ cũng như lưu trữ thông tin hồ sơ người dùng.
- **Dịch vụ sản phẩm:** Dịch vụ này liên quan đến việc quản lý sản phẩm có sẵn để bán trên trang web, chẳng hạn như thêm sản phẩm mới, cập nhật sản phẩm, xoá sản phẩm hay xem chi tiết sản phẩm
- **Dịch vụ tìm kiếm:** Dịch vụ này liên quan đến việc tìm kiếm các sản phẩm trên trang web để xem chi tiết, thêm giỏ hàng hoặc mua sắm. Có thể tìm kiếm bằng text.

- **Dịch vụ giỏ hàng:** Dịch vụ này có khả năng chịu trách nhiệm quản lý các chức năng giỏ hàng của trang web, chẳng hạn như cho phép người dùng thêm các mặt hàng vào giỏ hàng của họ, cập nhật giỏ hàng khi các mặt hàng được thêm vào hoặc xoá và tính toán tổng chi phí của đơn hàng.
- **Dịch vụ đặt hàng:** Dịch vụ này liên quan đến quản lý quy trình đặt hàng và thực hiện đơn hàng, chẳng hạn như nhận và xử lý đơn hàng, theo dõi mức tồn kho và cập nhật cho khách hàng về trạng thái đơn hàng.
- **Dịch vụ thanh toán:** Dịch vụ này sẽ xử lý quá trình thanh toán, chẳng hạn như chấp nhận thanh toán bằng tài khoản online (ví điện tử), xác minh tính hợp lệ của thông tin thanh toán và bắt đầu giao dịch với cổng thanh toán hoặc ngân hàng.
- **Dịch vụ vận chuyển:** Dịch vụ này sẽ quản lý việc vận chuyển và giao sản phẩm cho khách hàng, chẳng hạn như theo dõi các gói hàng, phối hợp với hãng vận chuyển và tính toán chi phí vận chuyển.
- **Dịch vụ nhân viên:** Dịch vụ này có thể liên quan đến việc quản lý tài khoản và quyền của nhân viên, chẳng hạn như cho phép nhân viên truy cập vào một số phần nhất định của trang web và theo dõi chỉ số hiệu suất của nhân viên.

❖ Biểu đồ phân rã



- Trong biểu đồ trên:
  - Mỗi hình chữ nhật đại diện cho một dịch vụ trong hệ thống (ví dụ: User Service, Product Service, etc.).
  - Mũi tên <> biểu thị tương tác giữa các dịch vụ. Ví dụ, User Service có thể tương tác với Cart Service để thêm sản phẩm vào giỏ hàng.

- Các dịch vụ có thể giao tiếp với nhau theo nhiều cách, như gửi yêu cầu HTTP, sử dụng message queue, hoặc các cách khác tùy thuộc vào cấu trúc và yêu cầu của hệ thống.

4. **Trình bày các dạng communication giữa các service với nhau (syschronous và asynchronous) với code và ví dụ. Cập nhật và bổ sung từ Bài tập 3.**

❖ **Các dạng communication giữa các service syschronous:**

- HTTP/HTTPS Request-Response: Trong giao thức HTTP, dịch vụ gửi một yêu cầu (request) đến một dịch vụ khác và đợi cho đến khi nhận được một phản hồi (response) trước khi tiếp tục thực hiện. Quá trình này là đồng bộ vì dịch vụ gửi yêu cầu phải chờ cho đến khi nhận được phản hồi.
  Ví dụ:

```
import requests

def make_request():
    response = requests.get('https://api.example.com/data')
    return response.text
```

- Remote Procedure Call (RPC): RPC là một phương thức giao tiếp mà một dịch vụ gửi một yêu cầu cho một dịch vụ khác để thực thi một hàm hoặc phương thức cụ thể và chờ đợi kết quả trước khi tiếp tục thực hiện. Quá trình này cũng là đồng bộ vì dịch vụ gọi trực tiếp chờ đợi cho đến khi hàm hoặc phương thức được thực thi và kết quả được trả về.
  Ví dụ:

```
import xmlrpc.client

def rpc_call():
    proxy = xmlrpc.client.ServerProxy("http://localhost:8000/")
    result = proxy.add(4, 5)
    return result
```

- WebSocket Communication: Trong môi trường WebSocket, dịch vụ có thể thiết lập một kết nối hai chiều và gửi và nhận dữ liệu một cách liên tục. Mặc dù WebSocket cho phép giao tiếp real-time, nhưng quá trình gửi và nhận dữ liệu vẫn là đồng bộ.
  Ví dụ:

```python
import websocket

def websocket_communication():
    ws = websocket.create_connection("ws://localhost:8000/")
    ws.send("Hello, server!")
    result = ws.recv()
    ws.close()
    return result
```

❖ **Các dạng communication giữa các service asynchronous:**

- Message Queues: Trong hệ thống hàng đợi tin nhắn, dịch vụ gửi tin nhắn đến một hàng đợi mà không cần chờ đợi cho đến khi tin nhắn được xử lý. Dịch vụ khác có thể nhận và xử lý các tin nhắn từ hàng đợi một cách không đồng bộ, có nghĩa là chúng có thể tiếp tục thực hiện các công việc khác trong khi đang chờ đợi tin nhắn.

Ví dụ:

```python
import pika

def send_message_to_queue():
    connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
    channel = connection.channel()
    channel.queue_declare(queue='task_queue')
    channel.basic_publish(exchange='',
                          routing_key='task_queue',
                          body='Hello, queue!')
    connection.close()

def receive_message_from_queue():
    connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
    channel = connection.channel()
    method_frame, header_frame, body = channel.basic_get(queue='task_queue')
    if method_frame:
        channel.basic_ack(method_frame.delivery_tag)
        return body.decode('utf-8')
    else:
        return None
    connection.close()
```

- Event Streams: Trong một môi trường dữ liệu dòng sự kiện, dịch vụ có thể gửi và nhận các sự kiện một cách không đồng bộ. Các sự kiện có thể được gửi và xử lý ngay khi chúng xảy ra mà không cần chờ đợi cho đến khi dịch vụ khác phản hồi.
  Ví dụ:

```
import asyncio

async def event_stream_consumer():
    while True:
        event = await get_event_from_stream()
        process_event(event)

async def get_event_from_stream():
    # Code to fetch event from stream asynchronously
    pass

def process_event(event):
    # Code to process event
    pass
```

- Callback Functions: Trong một số trường hợp, các dịch vụ có thể sử dụng callback functions để thực hiện giao tiếp không đồng bộ. Dịch vụ gửi yêu cầu và chỉ định một hàm callback để xử lý kết quả khi nó được trả về.
  Ví dụ:

```
import requests

def make_async_request(callback):
    response = requests.get('https://api.example.com/data')
    callback(response.text)

def process_response(response):
    # Code to process response
    pass

make_async_request(process_response)
```

5. **Trình bày sử dụng các dạng communication giữa các service với code cho hệ ecomSys.**

❖ Giao tiếp giữa manager và book trong update book

```python
class UpdateBookView(APIView):
    def put(self, request, book_id):
        token_verification_url = "http://localhost:4001/api/ecomSys/manager/verify-token/"
        headers = {'Authorization': request.headers.get('Authorization')}
        response = requests.get(token_verification_url, headers=headers)

        if response.status_code == 200:
            try:
                book = Book.objects.get(book_id=book_id)
            except Book.DoesNotExist:
                return Response({'error': 'Book not found'}, status=status.HTTP_404_NOT_FOUND)
            serializer = UpdateBookSerializer(book, data=request.data, context={'request': request})
            if serializer.is_valid():
                serializer.save()
                return Response(serializer.data, status=status.HTTP_200_OK)
            return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
        return Response({'error': 'Invalid token.'}, status=status.HTTP_401_UNAUTHORIZED)
```

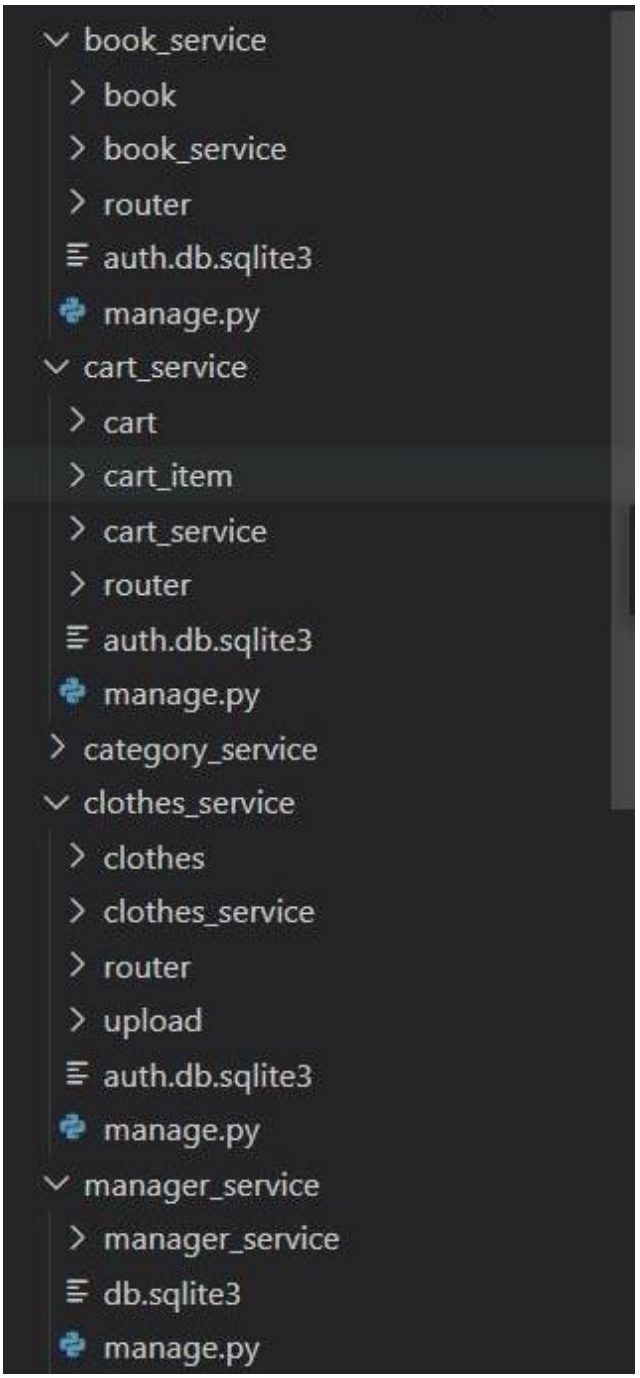
❖ Giao tiếp giữa manager và category trong delete category

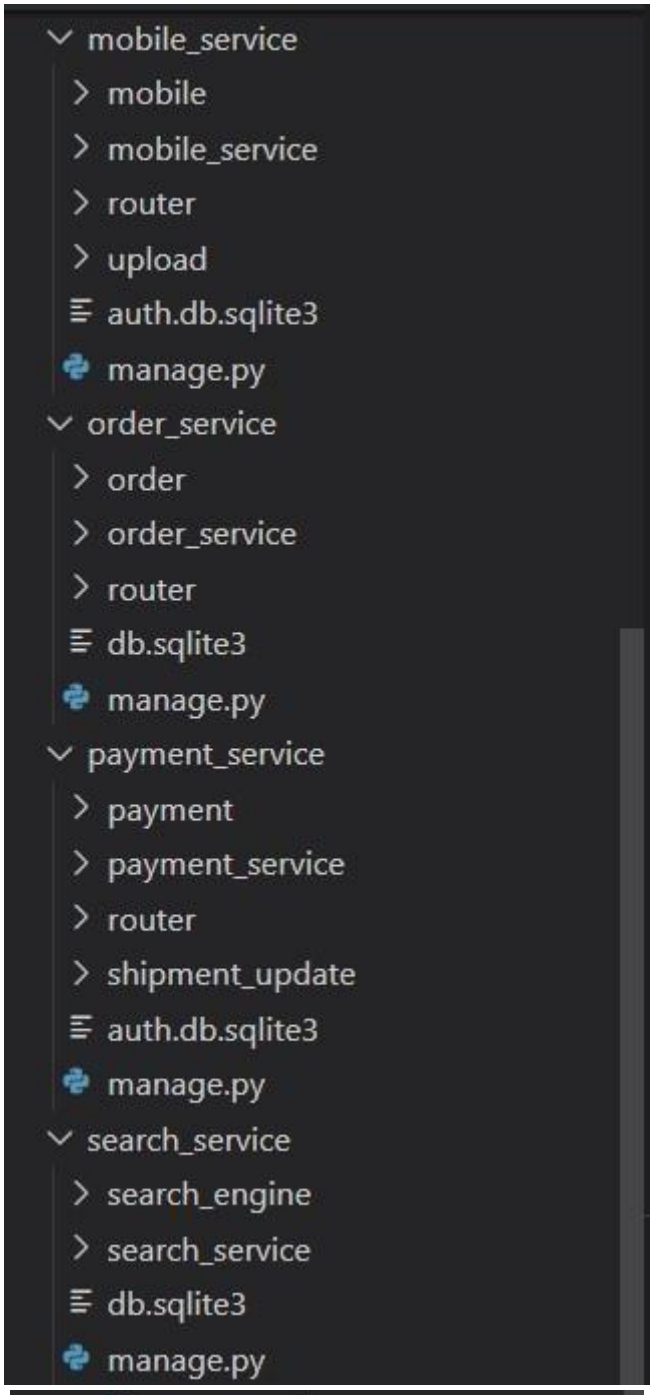
```python
class DeleteCategory(APIView):
    def delete(self, request, category_id):
        token_verification_url = "http://localhost:4001/api/ecomSys/manager/verify-token/"
        headers = {'Authorization': request.headers.get('Authorization')}
        response = requests.get(token_verification_url, headers=headers)

        if response.status_code == 200:
            try:
                category = Category.objects.get(category_id=category_id)
            except Category.DoesNotExist:
                return Response({'error': 'Category not found'}, status=status.HTTP_404_NOT_FOUND)

            serializer = CategorySerializer()
            serializer.destroy(category)

            return Response({'message': 'Category soft deleted'}, status=status.HTTP_204_NO_CONTENT)

        return Response({'error': 'Invalid token.'}, status=status.HTTP_401_UNAUTHORIZED)
```

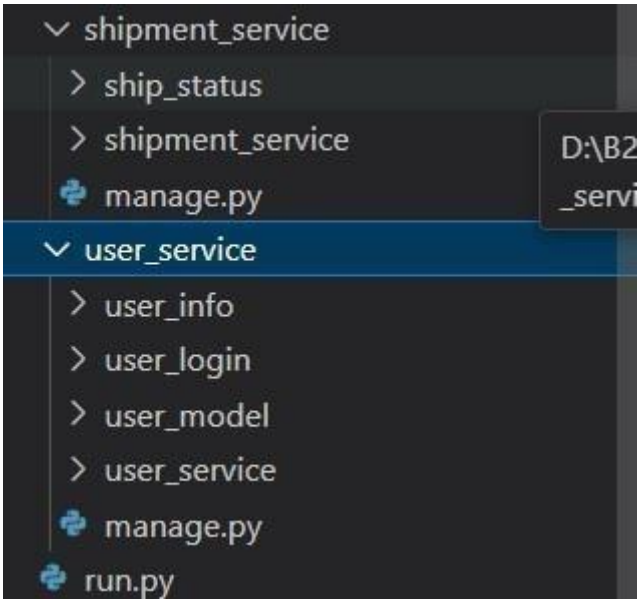
6. **Xây dựng biểu đồ activity cho tương tác giữ các services.**
7. **Xây dựng Biểu đồ lớp Phân tích cho từng service riêng lẻ.**
8. **Xây dựng data model cho từng service và công nghệ phát triển tương ứng (sử dụng cả 3 mySQL, PostgreSQL, mongoDB).**
9. **Xây dựng biểu đồ lớp thiết kế và kiến trúc cho từng service với MVT Django.**
10. **Code và demo.**

Project: ecomSys_cnpm2.03_Do_Duc_Thu

1. 10 services in Django: user, manager, cart, order, search, book, mobile, clothes, shipment, payment



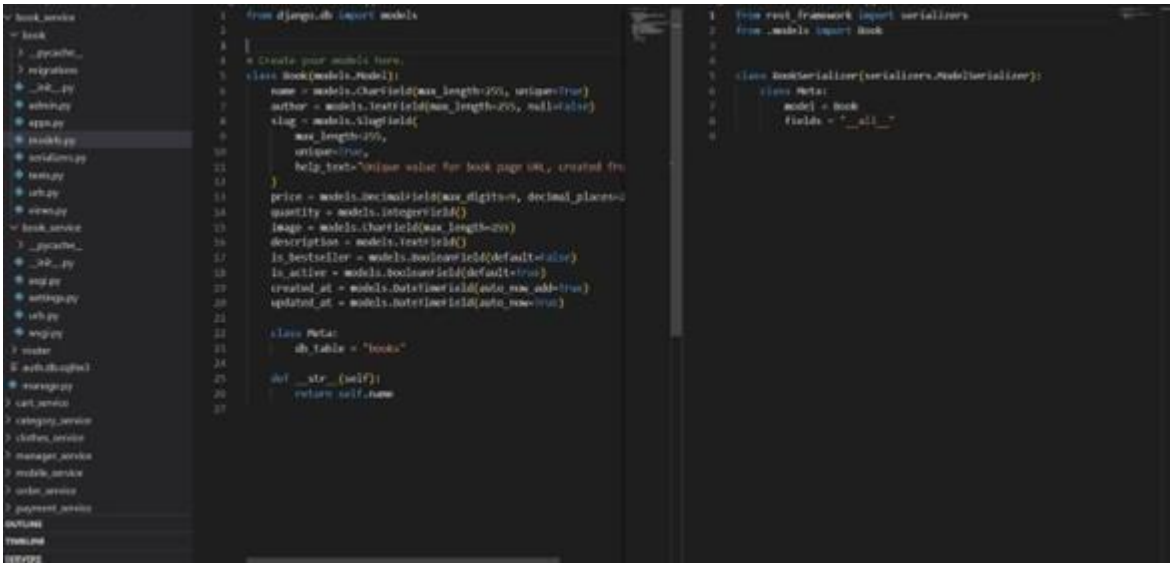

mobile_service
mobile
mobile_service
router
upload
auth.db.sqlite3
manage.py
order_service
order
order_service
router
db.sqlite3
manage.py
payment_service
payment
payment_service
router
shipment_update
auth.db.sqlite3
manage.py
search_service
search_engine
search_service
db.sqlite3
manage.py

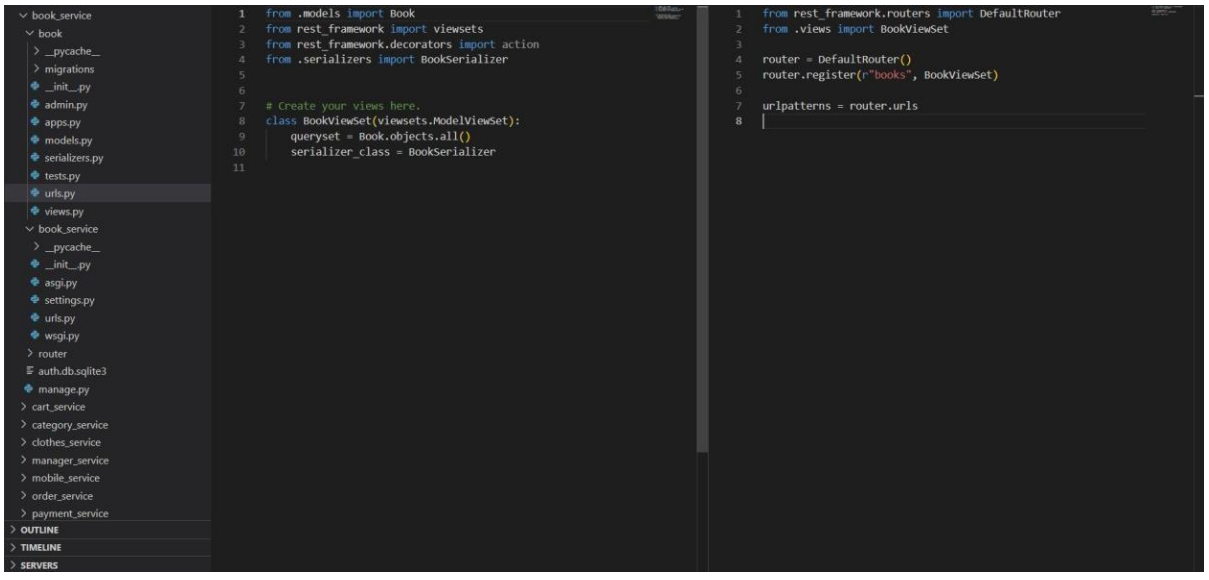

shipment_service
ship_status
shipment_service
manage.py
user_service
user_info
user_login
user_model
user_service
manage.py
run.py

2. Rest API to connect to the necessary services

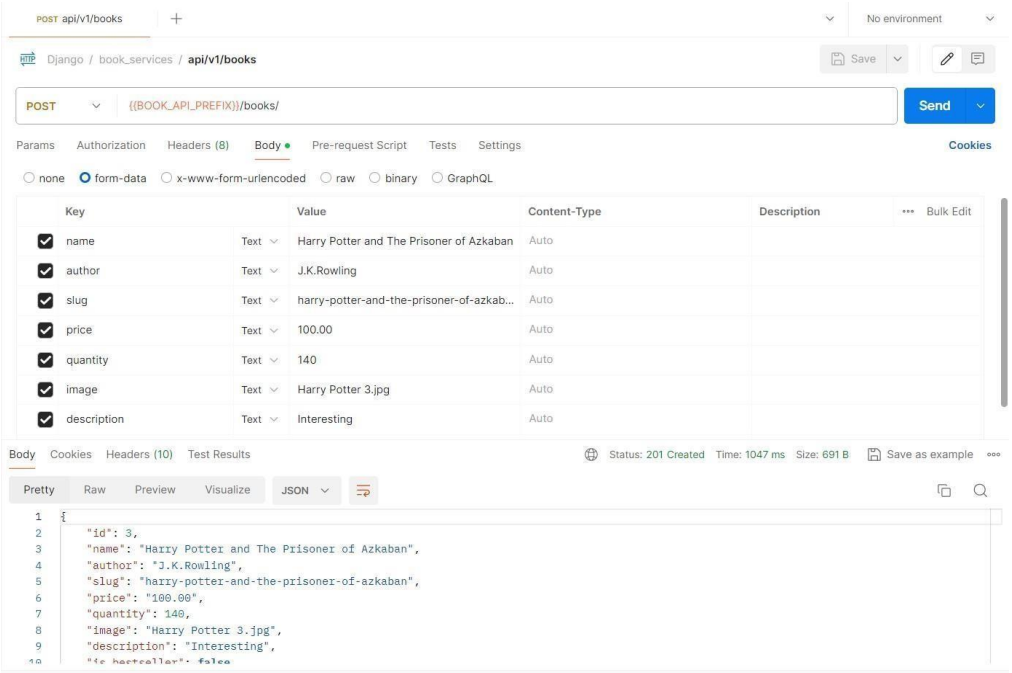
+ book_service:

```
DATABASES = {
    'default': {},
    'auth_db':{
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'auth.db.sqlite3',
    },
    'mongodb': {
        'ENGINE' : 'djongo',
        'NAME': 'book_service',
    }
}

DATABASE_ROUTERS = ['router.database_routers.AuthDBRouter', 'router.database_routers.MongoDBRouter']
```

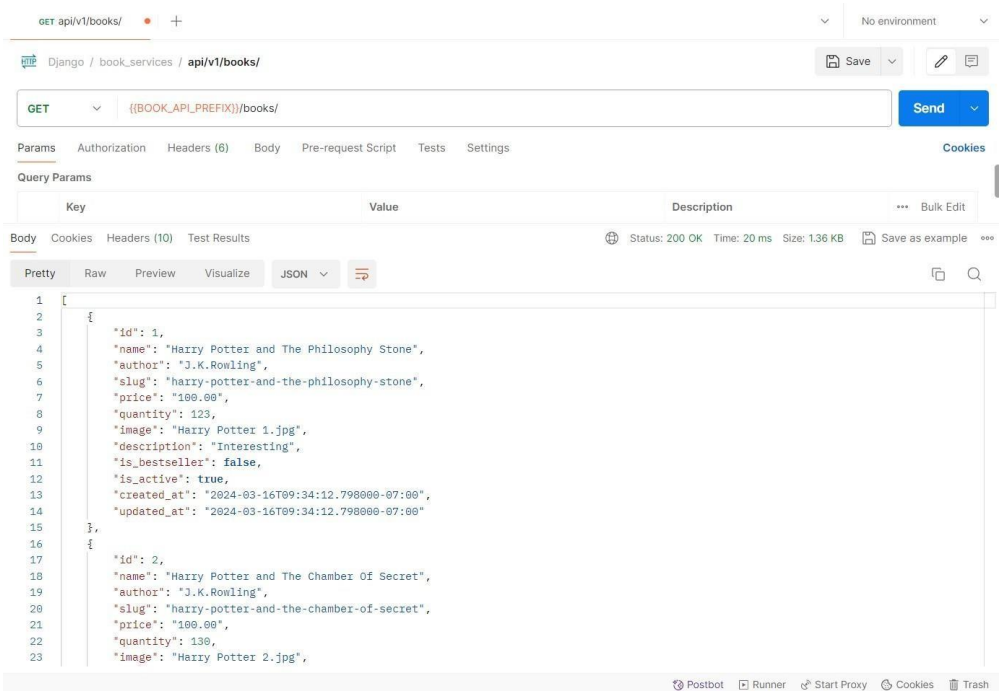




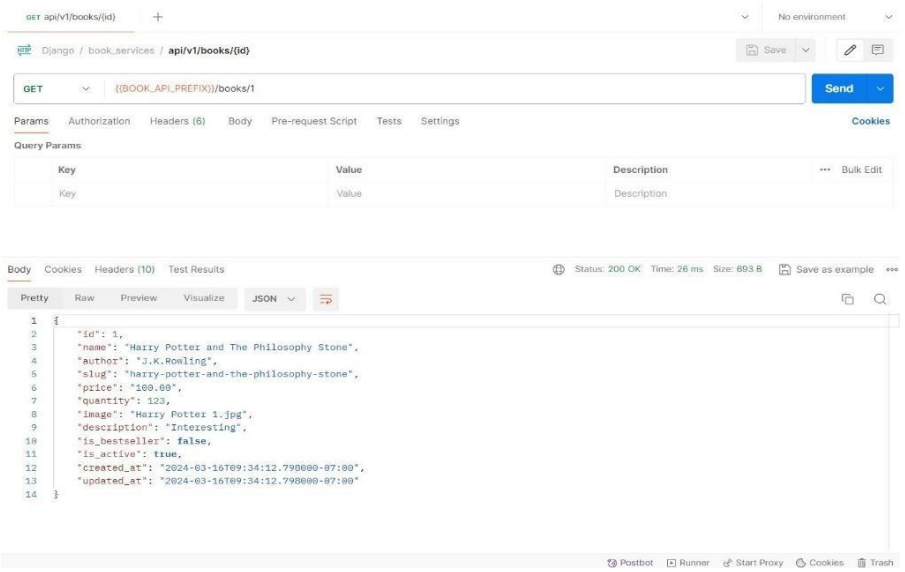
- API to create book



- API to get all books:

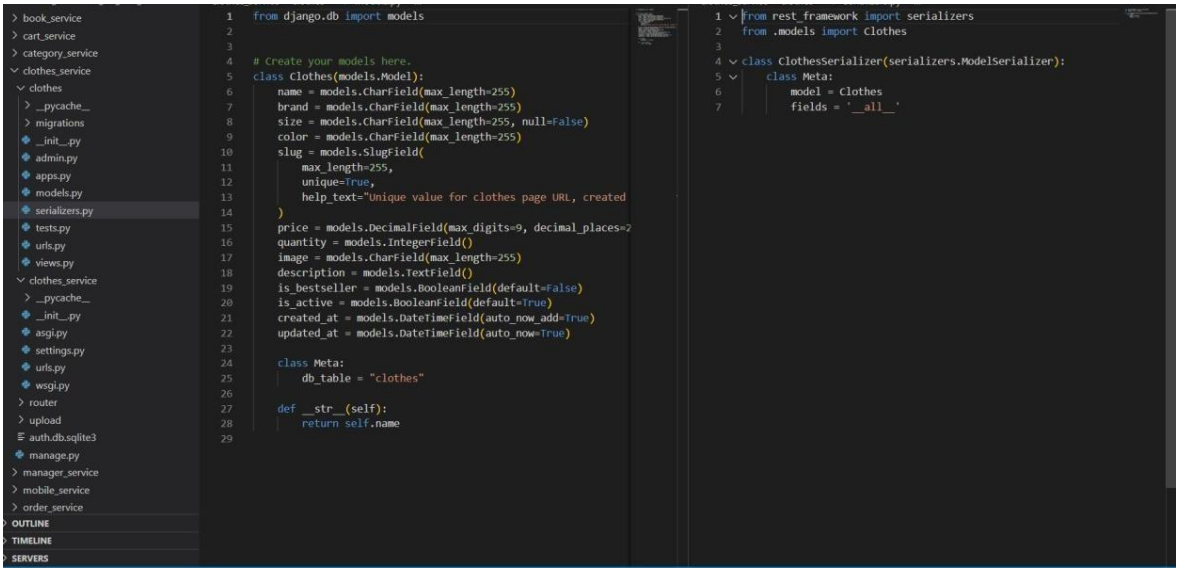


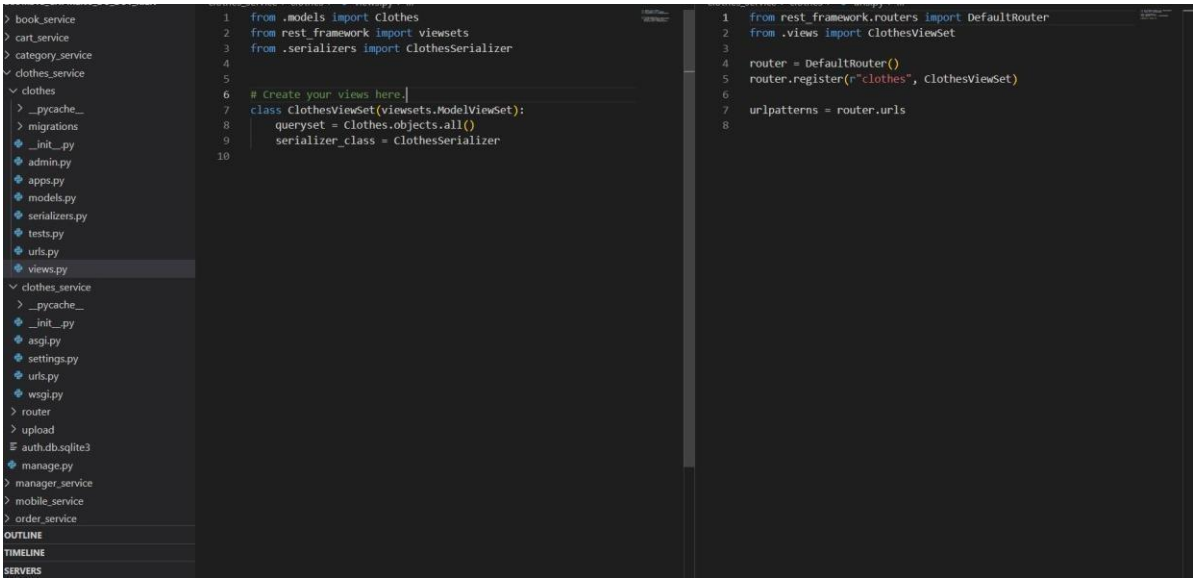
- API to get book by ID:



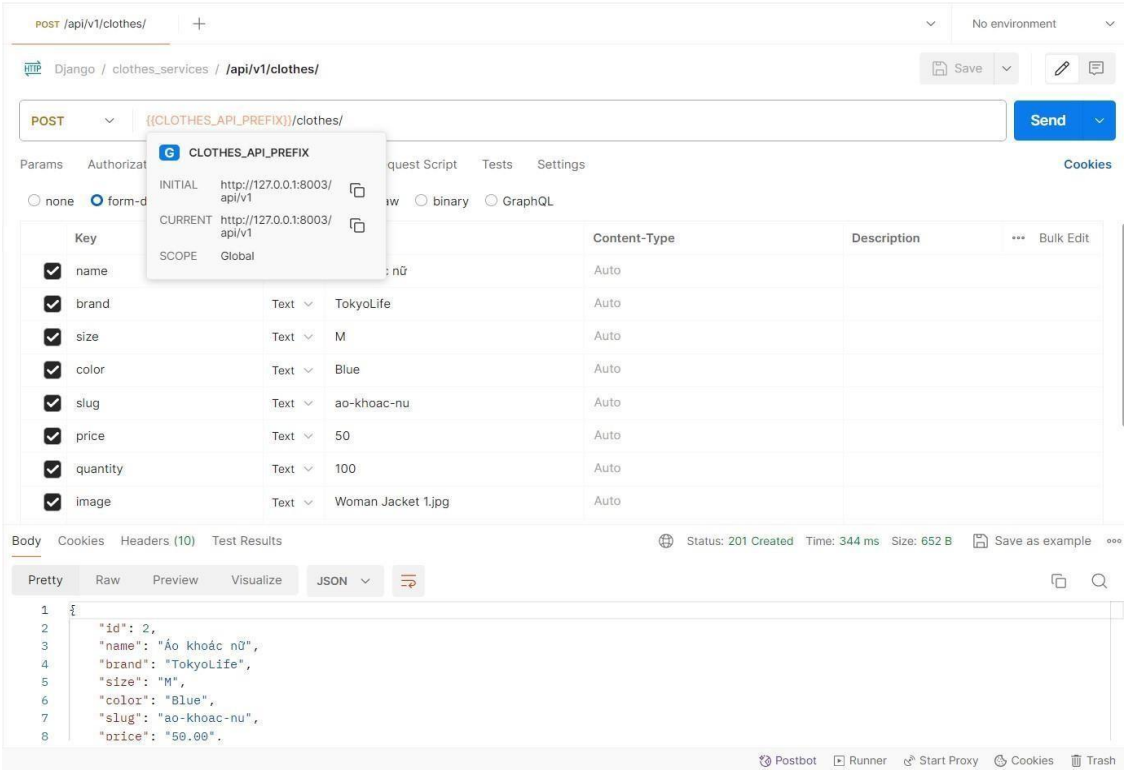
+ clothes_service:

```python
DATABASES = {
    'default': {},
    'auth_db':{
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'auth.db.sqlite3',
    },
    'mongodb': {
        'ENGINE' : 'djongo',
        'NAME': 'clothes_service',
    },
}

DATABASE_ROUTERS = ['router.database_routers.AuthDBRouter', 'router.database_routers.MongoDBRouter']

```

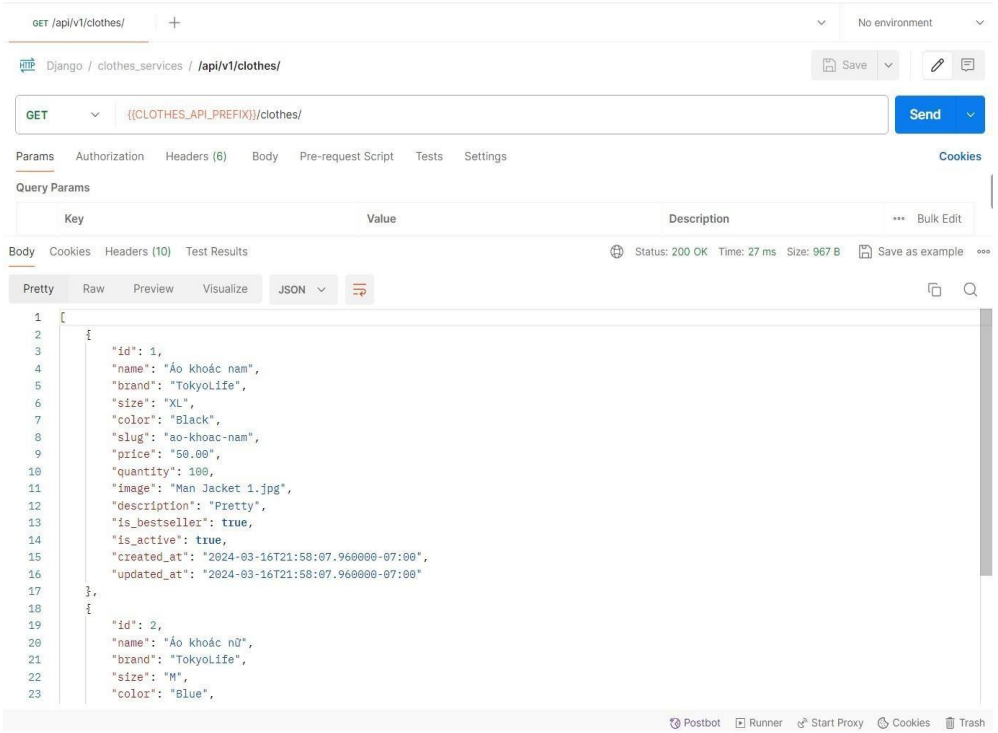




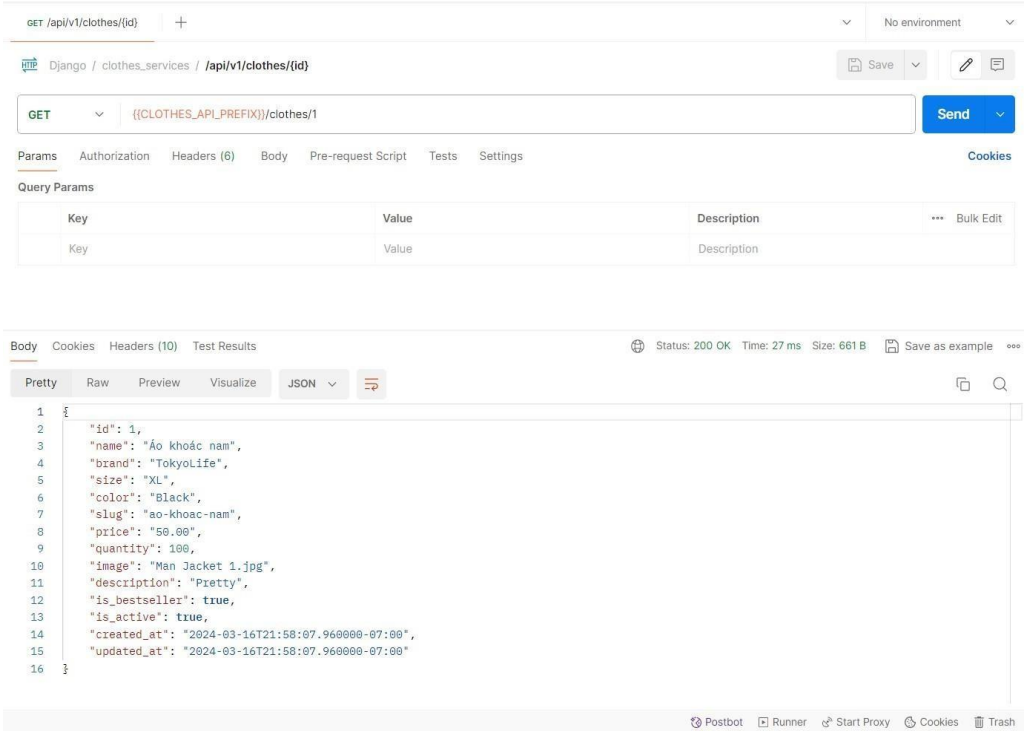
- API to create clothes:



- API to get all clothes:

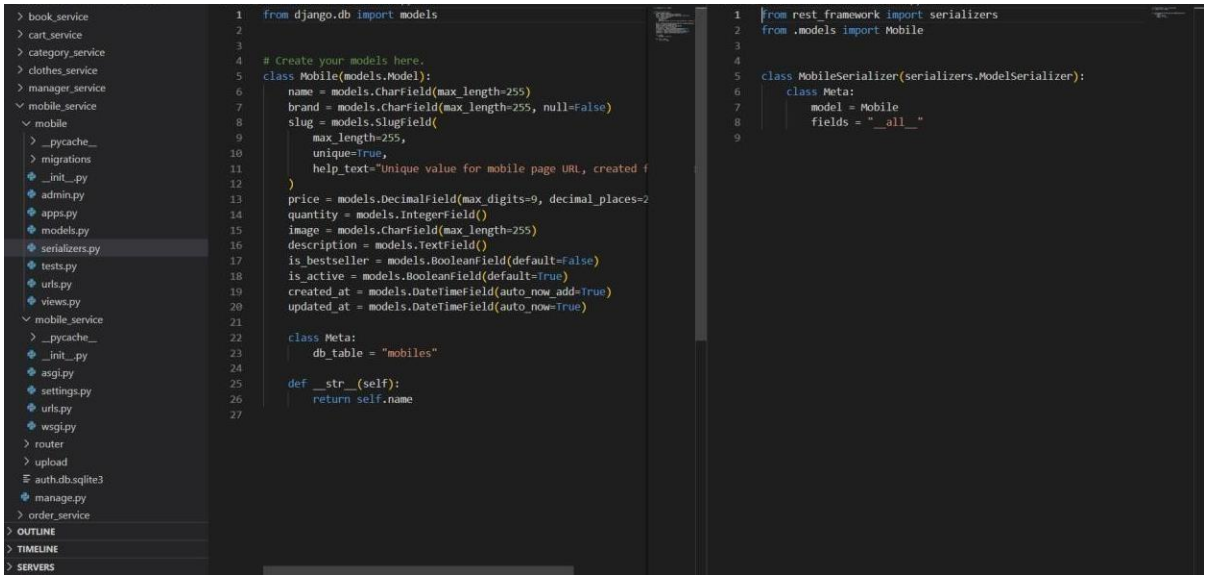


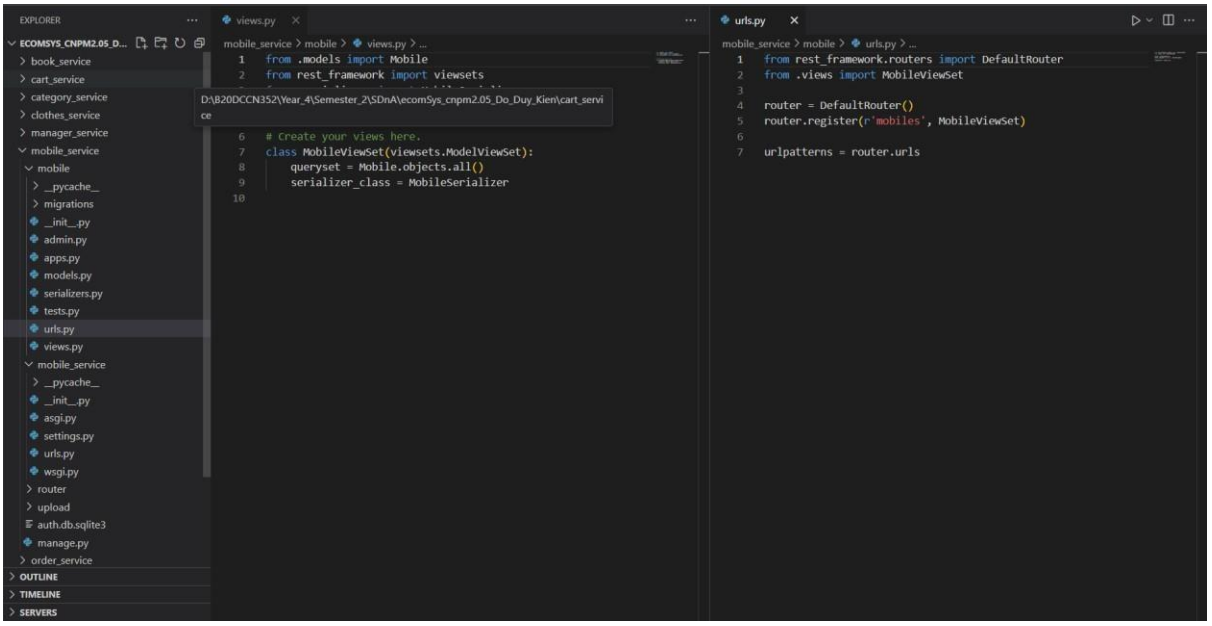
- API to get clothes by ID:



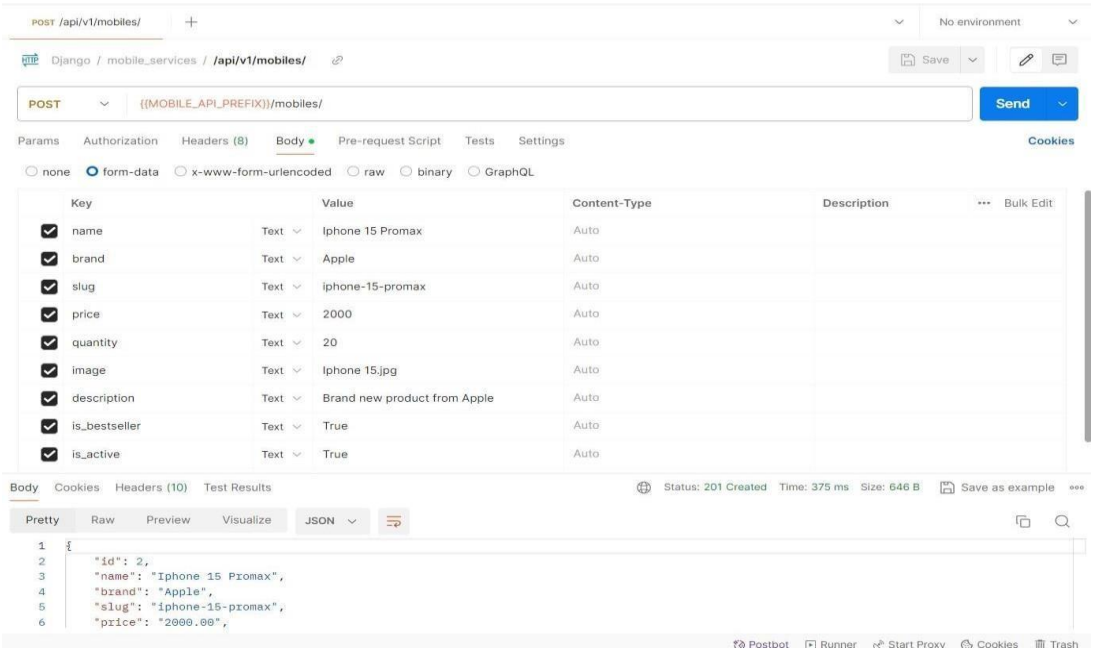
+ mobile_service:

```
DATABASES = {
    'default': {},
    'auth_db':{
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'auth.db.sqlite3',
    },
    'mongodb': {
        'ENGINE' : 'djongo',
        'NAME': 'mobile_service',
    },
}

DATABASE_ROUTERS = ['router.database_routers.AuthDBRouter', 'router.database_routers.MongoDBRouter']

```

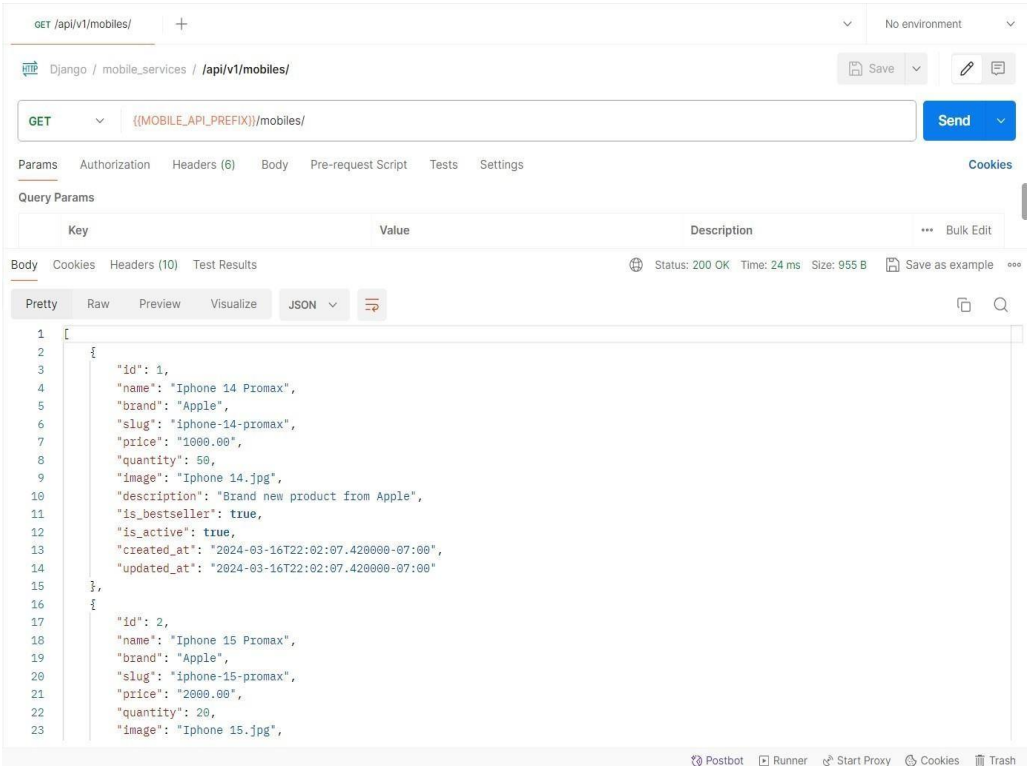




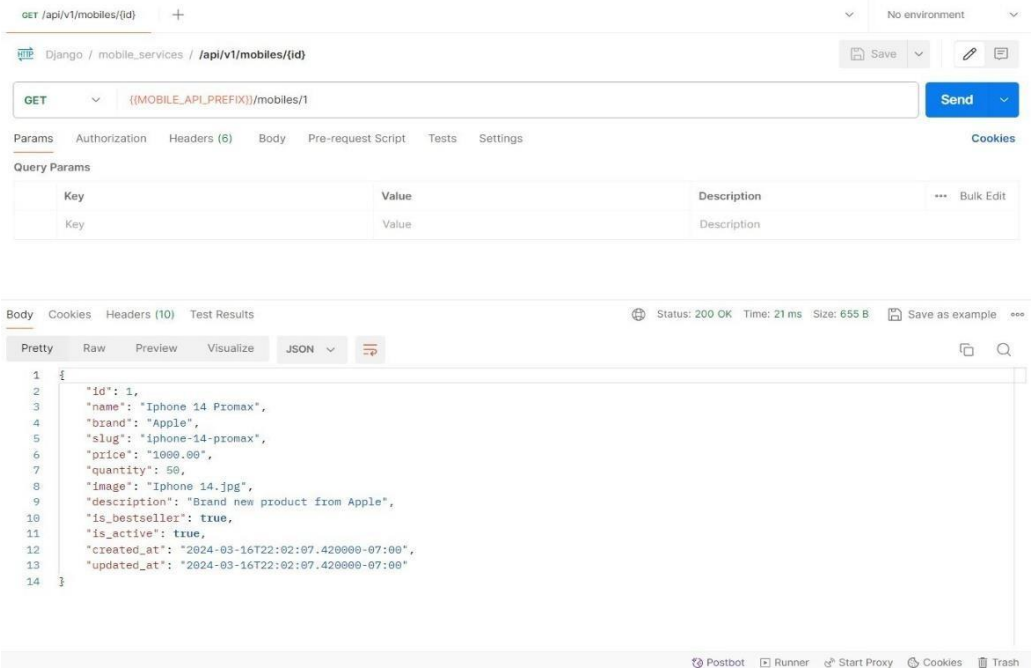
- API to create mobile:



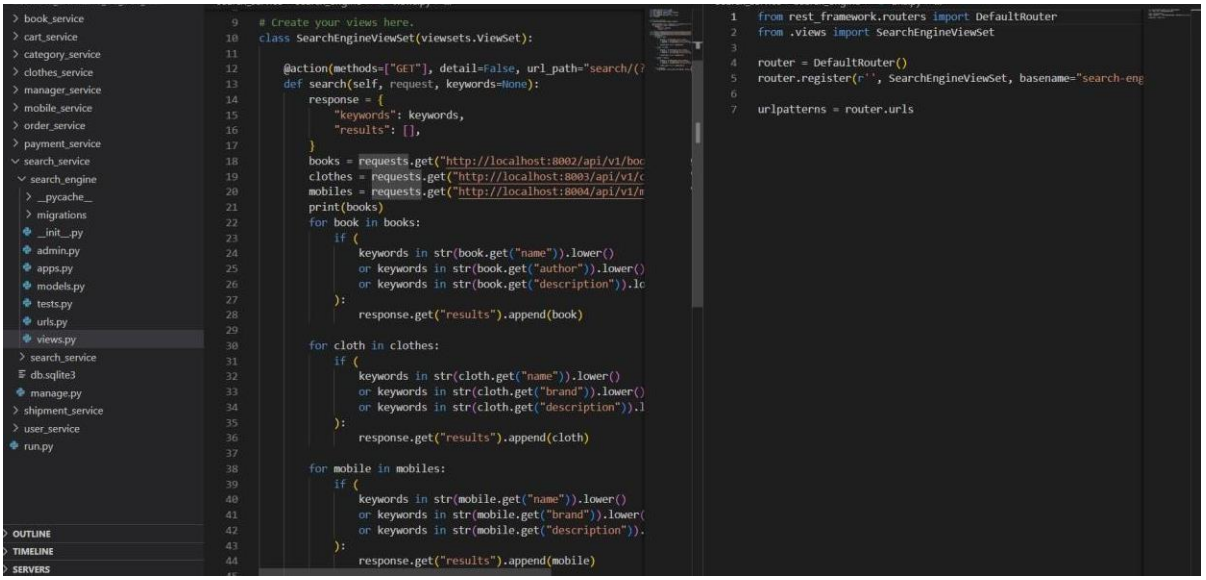
- API to get all mobiles:



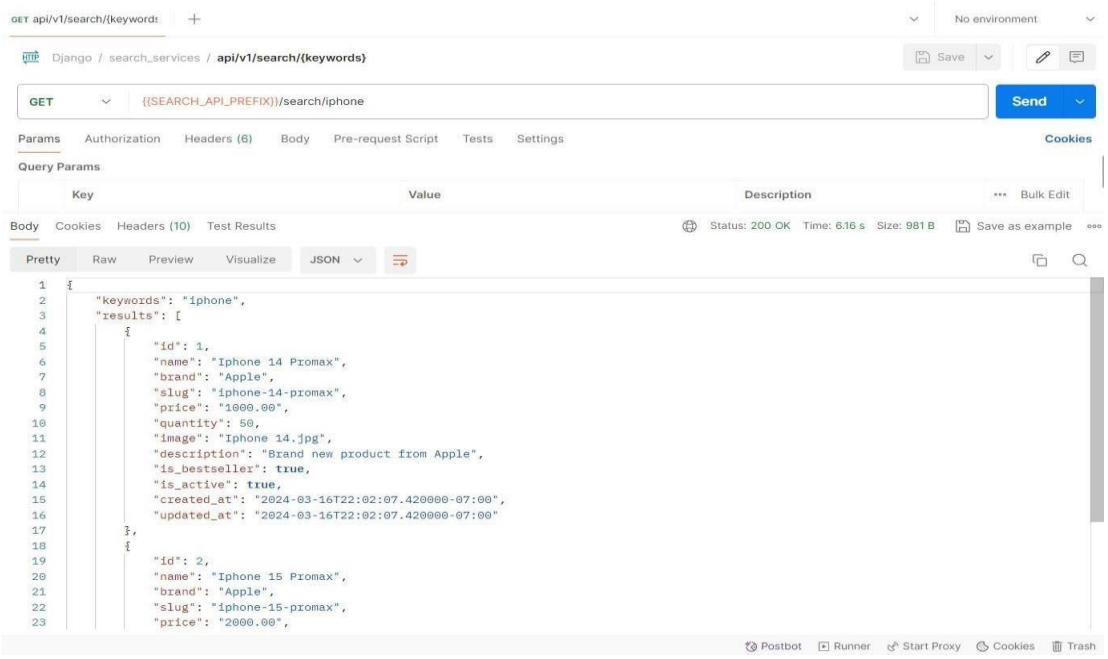
- API to get mobile by ID:



+ search_service:

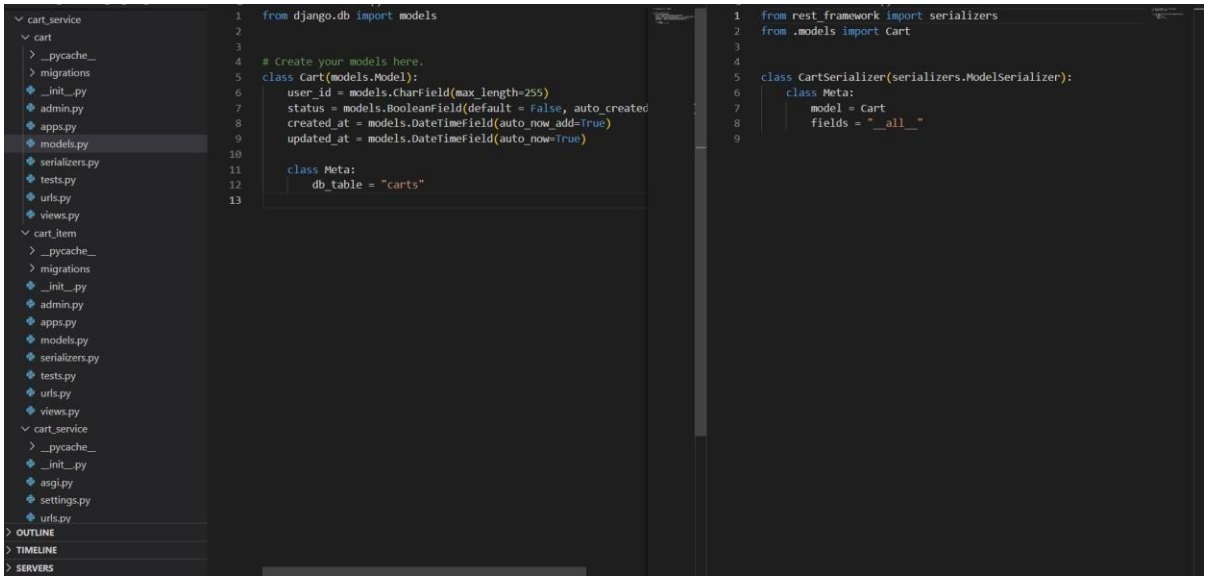


- API to search products:



+ cart_service:

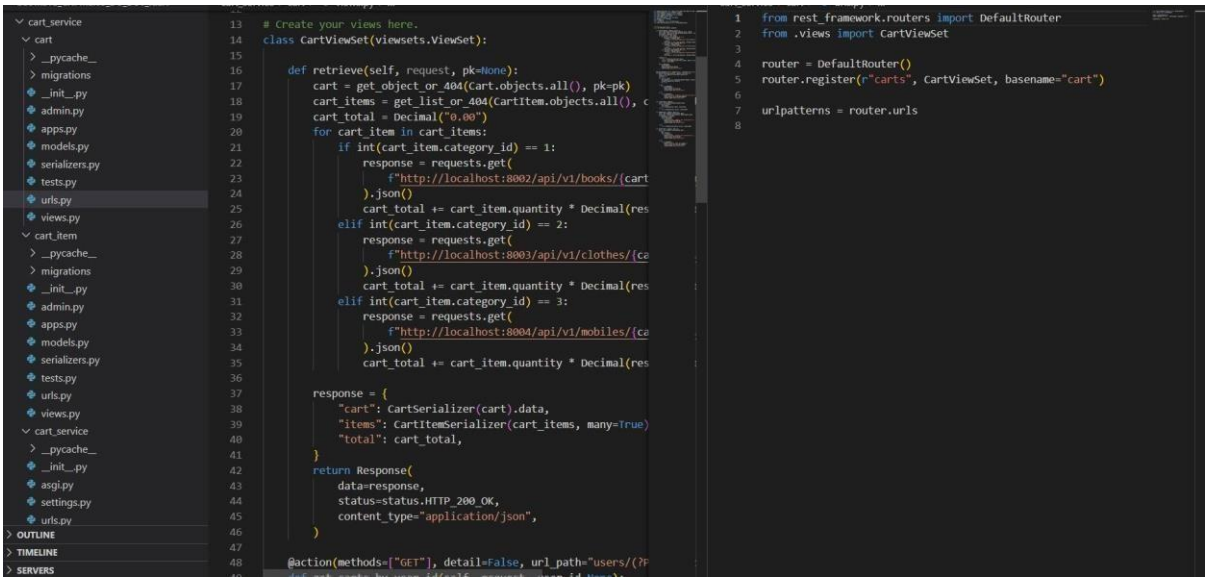
- cart:



```
cart_service
cart
__pycache__
migrations
__init__.py
admin.py
apps.py
models.py
serializers.py
tests.py
urls.py
views.py
cart_item
__pycache__
migrations
__init__.py
admin.py
apps.py
models.py
serializers.py
tests.py
urls.py
views.py
cart_service
__pycache__
__init__.py
asgi.py
settings.py
urls.py
OUTLINE
TIMELINE
SERVERS
```

```python
from django.db import models


# Create your models here.
class Cart(models.Model):
    user_id = models.CharField(max_length=255)
    status = models.BooleanField(default = False, auto_created
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    class Meta:
        db_table = "carts"

```

```python
from rest_framework import serializers
from .models import Cart


class CartSerializer(serializers.ModelSerializer):
    class Meta:
        model = Cart
        fields = "__all__"

```

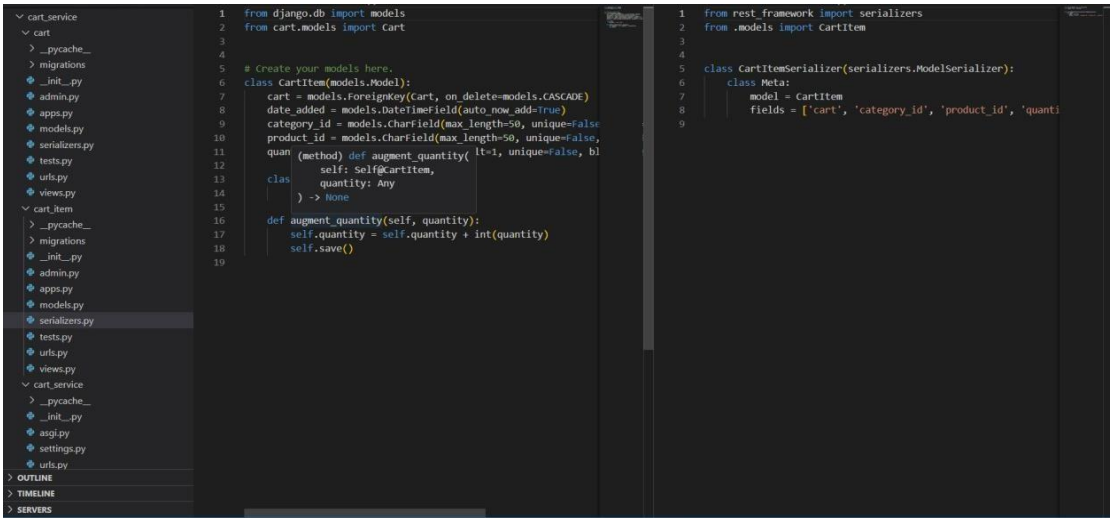


```
cart_service
cart
__pycache__
migrations
__init__.py
admin.py
apps.py
models.py
serializers.py
tests.py
urls.py
views.py
cart_item
__pycache__
migrations
__init__.py
admin.py
apps.py
models.py
serializers.py
tests.py
urls.py
views.py
cart_service
__pycache__
__init__.py
asgi.py
settings.py
urls.py
OUTLINE
TIMELINE
SERVERS
```

```python
# Create your views here.
class CartViewSet(viewsets.ViewSet):

    def retrieve(self, request, pk=None):
        cart = get_object_or_404(Cart.objects.all(), pk=pk)
        cart_items = get_list_or_404(CartItem.objects.all(), c
        cart_total = Decimal("0.00")
        for cart_item in cart_items:
            if int(cart_item.category_id) == 1:
                response = requests.get(
                    f"http://localhost:8002/api/v1/books/{cart
                ).json()
                cart_total += cart_item.quantity * Decimal(res
            elif int(cart_item.category_id) == 2:
                response = requests.get(
                    f"http://localhost:8003/api/v1/clothes/{ca
                ).json()
                cart_total += cart_item.quantity * Decimal(res
            elif int(cart_item.category_id) == 3:
                response = requests.get(
                    f"http://localhost:8004/api/v1/mobiles/{ca
                ).json()
                cart_total += cart_item.quantity * Decimal(res

        response = {
            "cart": CartSerializer(cart).data,
            "items": CartItemSerializer(cart_items, many=True)
            "total": cart_total,
        }
        return Response(
            data=response,
            status=status.HTTP_200_OK,
            content_type="application/json",
        )

    @action(methods=["GET"], detail=False, url_path="users/(?P
```

```python
from rest_framework.routers import DefaultRouter
from .views import CartViewSet

router = DefaultRouter()
router.register(r"carts", CartViewSet, basename="cart")

urlpatterns = router.urls

```

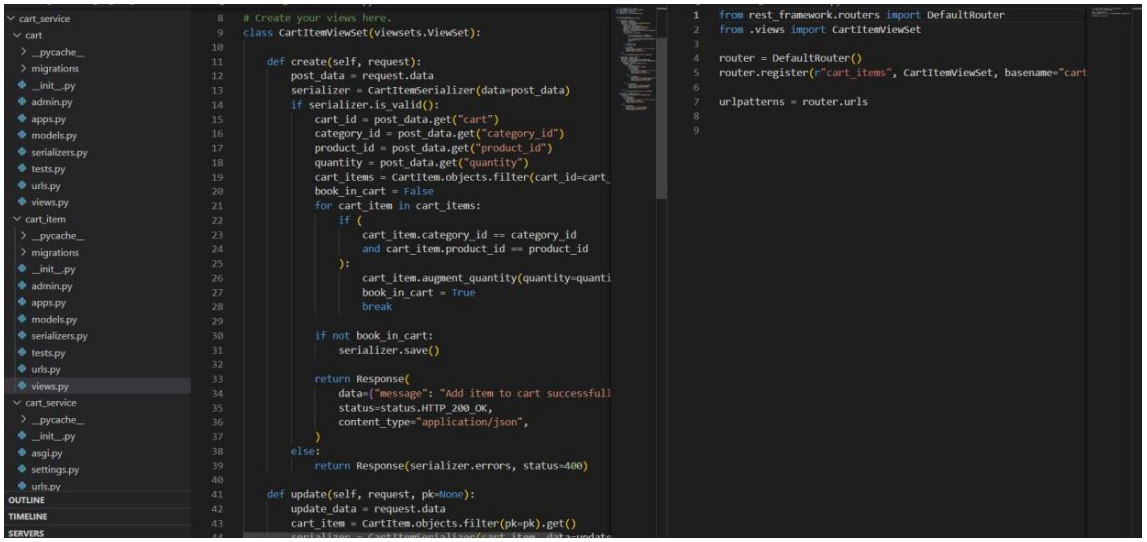
- cart_item:



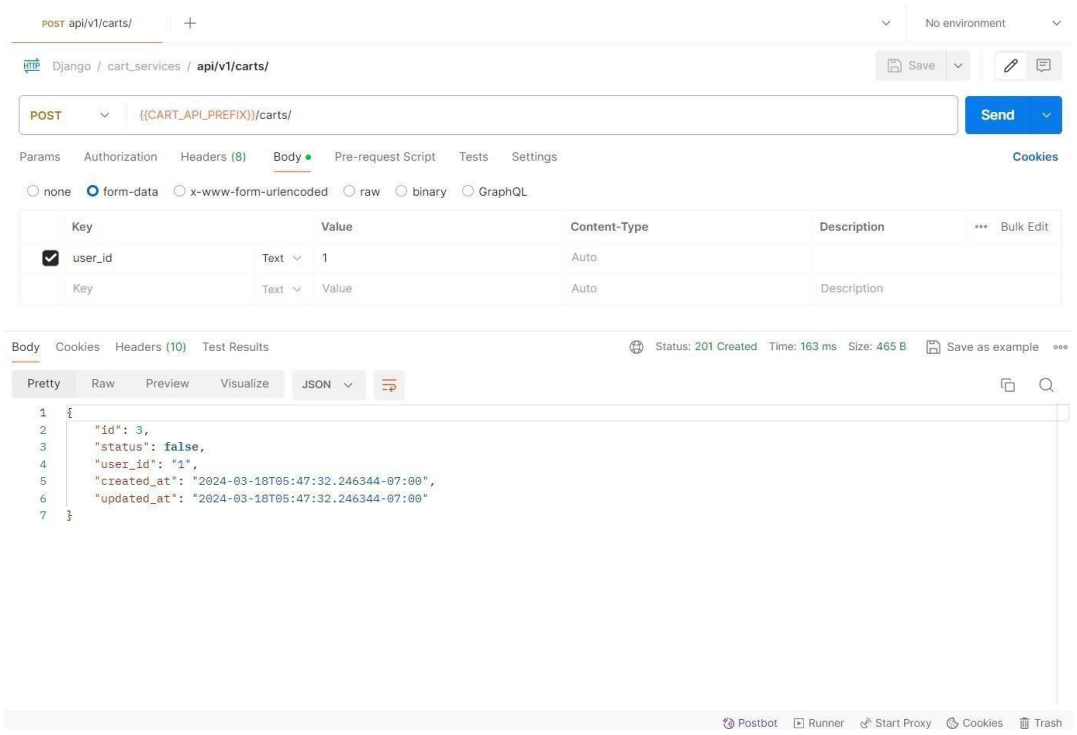
```
cart_service
cart
__pycache__
migrations
__init__.py
admin.py
apps.py
models.py
serializers.py
tests.py
urls.py
views.py
cart_item
__pycache__
migrations
__init__.py
admin.py
apps.py
models.py
serializers.py
tests.py
urls.py
views.py
cart_service
__pycache__
__init__.py
asgi.py
settings.py
urls.py
OUTLINE
TIMELINE
SERVERS
```

```python
from django.db import models
from cart.models import Cart


# Create your models here.
class CartItem(models.Model):
    cart = models.ForeignKey(Cart, on_delete=models.CASCADE)
    date_added = models.DateTimeField(auto_now_add=True)
    category_id = models.CharField(max_length=50, unique=False
    product_id = models.CharField(max_length=50, unique=False,
    quan (method) def augment_quantity( lt=1, unique=False, bl
        self: Self@CartItem,
        quantity: Any
    ) -> None
    cla

    def augment_quantity(self, quantity):
        self.quantity = self.quantity + int(quantity)
        self.save()

```

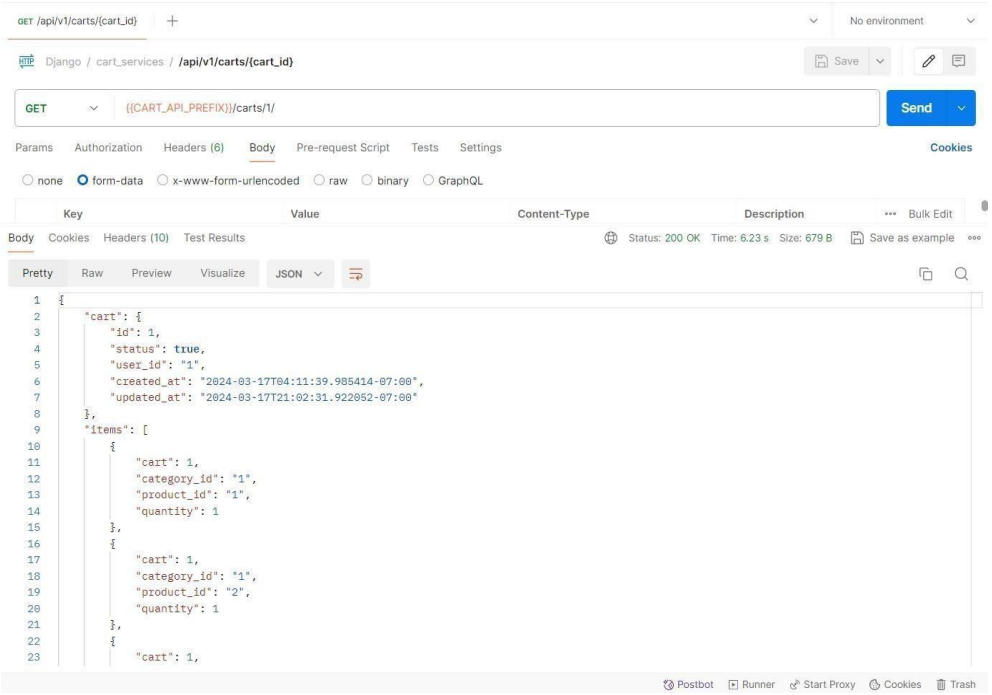
```python
from rest_framework import serializers
from .models import CartItem


class CartItemSerializer(serializers.ModelSerializer):
    class Meta:
        model = CartItem
        fields = ['cart', 'category_id', 'product_id', 'quanti

```



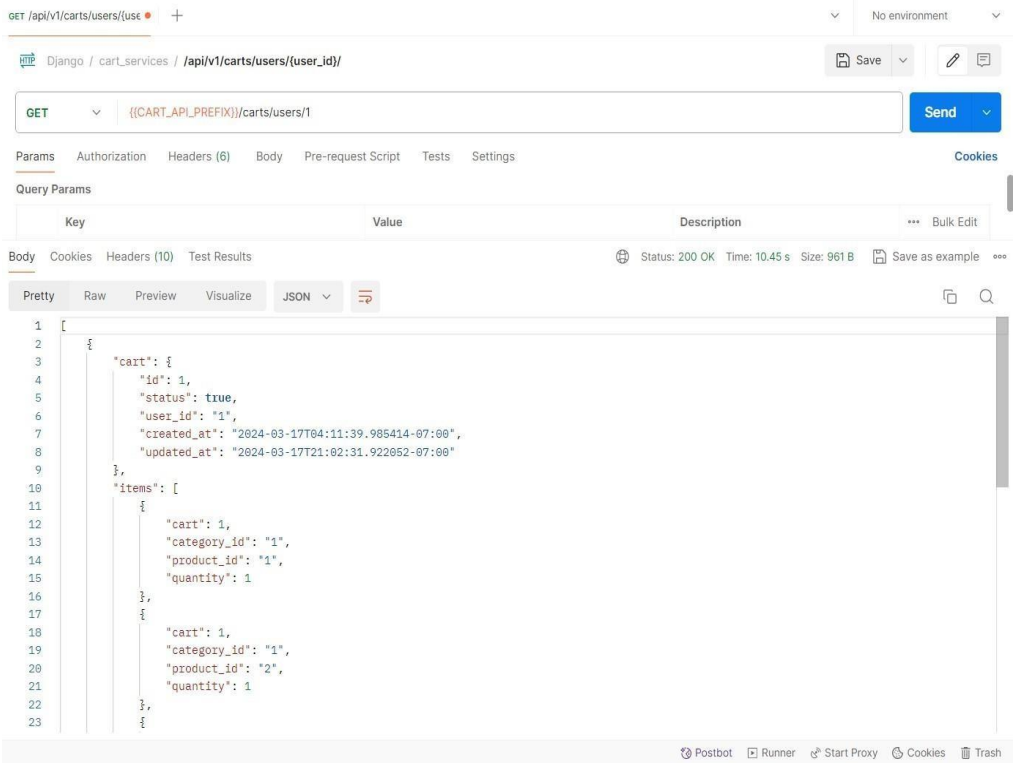
- API to create cart:



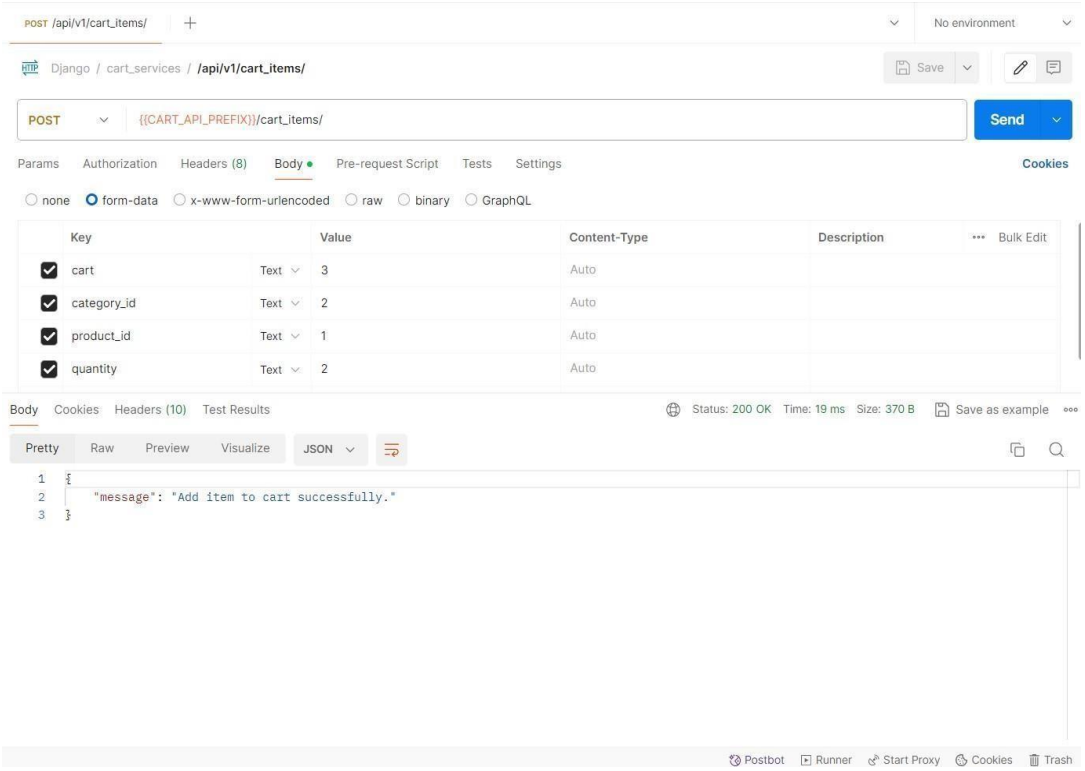
- API to get cart by ID:



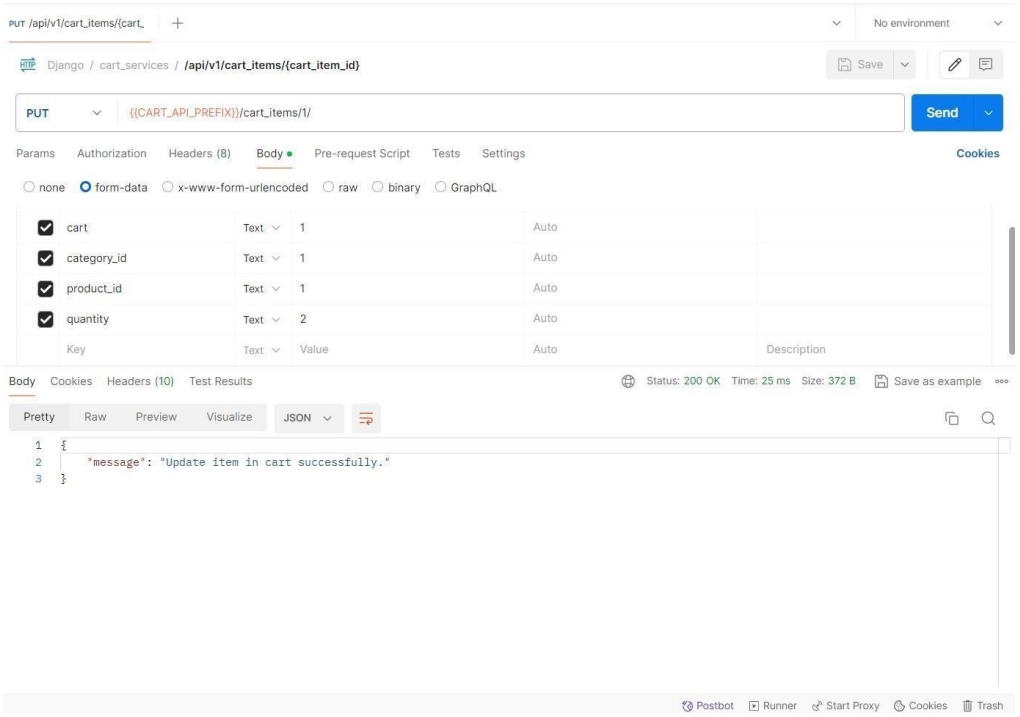
- API to get cart with user ID:



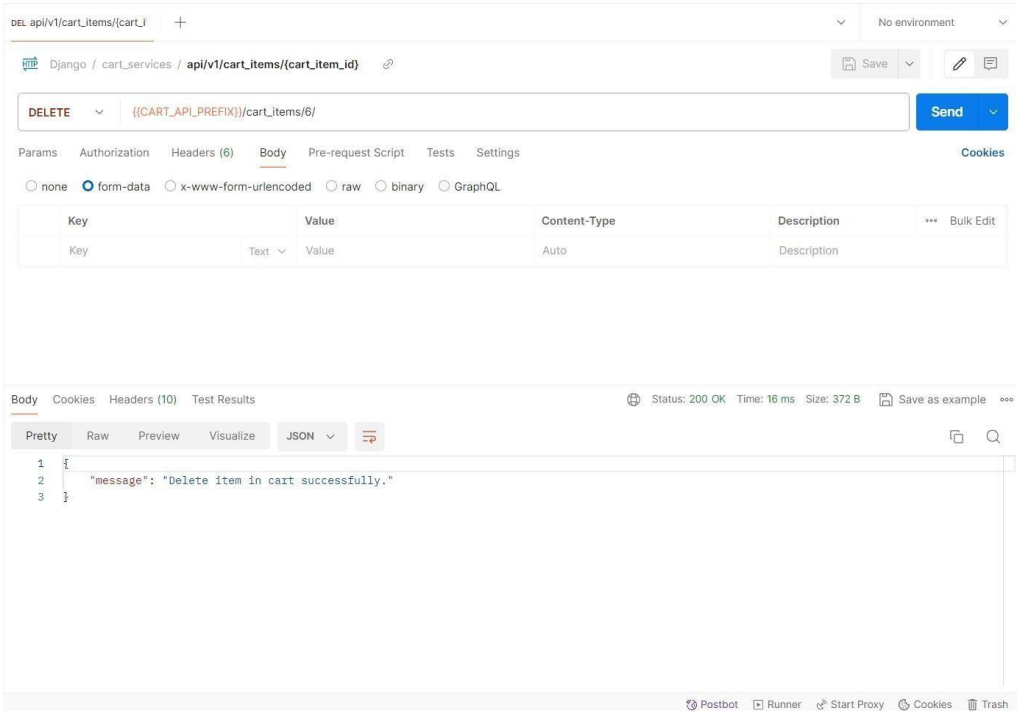
- API to add item to cart:



- API to update item in cart:



- API to delete item in cart:



3. Search process and Create cart, Add to cart

- Search process

views.py

search_service > search_engine > views.py > ...

```python
from django.shortcuts import render
from rest_framework import viewsets
from rest_framework.response import Response
from rest_framework import status
from rest_framework.decorators import action
import requests


# Create your views here.
class SearchEngineViewSet(viewsets.ViewSet):

    @action(methods=["GET"], detail=False, url_path="search/(?P<keywords>.+)")
    def search(self, request, keywords=None):
        response = {
            "keywords": keywords,
            "results": [],
        }
        books = requests.get("http://localhost:8002/api/v1/books/").json()
        clothes = requests.get("http://localhost:8003/api/v1/clothes/").json()
        mobiles = requests.get("http://localhost:8004/api/v1/mobiles/").json()
        print(books)
        for book in books:
            if (
                keywords in str(book.get("name")).lower()
                or keywords in str(book.get("author")).lower()
                or keywords in str(book.get("description")).lower()
            ):
                response.get("results").append(book)

        for cloth in clothes:
            if (
                keywords in str(cloth.get("name")).lower()
                or keywords in str(cloth.get("brand")).lower()
                or keywords in str(cloth.get("description")).lower()
            ):
                response.get("results").append(cloth)

```

```
        for mobile in mobiles:
            if (
                keywords in str(mobile.get("name")).lower()
                or keywords in str(mobile.get("brand")).lower()
                or keywords in str(mobile.get("description")).lower()
            ):
                response.get("results").append(mobile)

        return Response(
            data=response, status=status.HTTP_200_OK, content_type="application/json"
        )

```

- Create cart

```
def create(self, request):
    cart = CartSerializer(data=request.data)
    if cart.is_valid():
        cart.save()
        return Response(cart.data, status=201)
    else:
        return Response(cart.errors, status=400)
```

- Add to cart

```
def create(self, request):
    post_data = request.data
    serializer = CartItemSerializer(data=post_data)
    if serializer.is_valid():
        cart_id = post_data.get("cart")
        category_id = post_data.get("category_id")
        product_id = post_data.get("product_id")
        quantity = post_data.get("quantity")
        cart_items = CartItem.objects.filter(cart_id=cart_id)
        book_in_cart = False
        for cart_item in cart_items:
            if (
                cart_item.category_id == category_id
                and cart_item.product_id == product_id
            ):
                cart_item.augment_quantity(quantity=quantity)
                book_in_cart = True
                break

        if not book_in_cart:
            serializer.save()

        return Response(
            data={"message": "Add item to cart successfully."},
            status=status.HTTP_200_OK,
            content_type="application/json",
        )
    else:
        return Response(serializer.errors, status=400)
```